

Supporting Collaborative Tailoring

Helge Kahler

Ph. D. Thesis

Department of Communication, Journalism and Computer Science

Roskilde University, Denmark

August 2001

Summary

This dissertation addresses the support of *collaborative tailoring*, i. e. the technical and human art of modifying the functionality of software while the software is in use in the field and doing so together with others. This is an interesting and important issue for two reasons: firstly, software is rarely produced to be used by one person at one time, so most people will benefit from being able to change the software according to their needs. Secondly, since several people with similar tasks may or do tailor their software, there can be synergetic effects if they tailor collaboratively. The research question guiding the work described in this thesis is how collaborative tailoring can be adequately supported. The answer to this question is provided in the form of eight suggestions for collaborative tailoring. These suggestions include both technical and organizational aspects with a stress on the former.

The suggestions are: *provide objectification, allow sharing of tailoring files, allow browsing through tailoring files, provide awareness of tailoring activities, make annotations and automatic descriptions possible, allow for exploration of a tailoring file, make administration and coordination easy, and support a tailoring culture*. The thesis explains these suggestions and describes the research process. This process includes literature study and the subsequent development and usage of a prototypical software (a word processor extension and a groupware search tool) dealing with aspects of collaborative tailoring.

The dissertation consists of two parts. In the first part the various results and insights of the papers of the second part (collection of papers) are gathered, focused, and enhanced. The first part can be read as a contribution in its own right, where not only methodological issues are raised but also the process of generating suggestions for collaborative tailoring is described and discussed.

The seven papers of the collection contribute to the overall research question in different ways. The first and second paper shed a light on tailoring in general, present preliminary results on collaborative tailoring and highlight the necessity for more research about collaborative tailoring. The third, fifth, and sixth paper present the two cases word processor and groupware search tool. The fourth paper presents a method for usability

testing resulting from my work on the word processor case. The seventh paper presents the suggestions for collaborative tailoring derived from the previous work. A summary of each of the seven papers of the collection is provided in the introduction.

Resumé

Denne afhandling behandler understøttelsen af *collaborative tailoring* (samarbejde om tilpasning), dvs. de tekniske og menneskelige aspekter af at ændre softwarens funktionalitet, mens softwaren er i brug, og at gøre dette i samarbejde med andre. Dette er et interessant og vigtigt emne af to grunde: for det første er det sjældent, at software fremstilles til brug for én person én gang, så for de fleste mennesker vil det være en fordel at kunne ændre softwaren efter deres behov, og for det andet: eftersom mange mennesker med de samme opgaver kan eller vil skræddersy deres software, kan der opstå en synergieffekt, hvis de gør dette i fællesskab. Det forskningsspørgsmål, der har været styrende for det arbejde, der er beskrevet i denne afhandling er, hvordan “collaborative tailoring” adækvat kan understøttes. Svaret på dette spørgsmål gives i form af otte forslag til “collaborative tailoring”. Disse forslag omfatter både tekniske og organisatoriske aspekter med vægt på førstnævnte.

Forslagene er: *tilbyd objektivering, muliggør deling af tailoring-filer, tillad browsing af tailoring-filer, understøt opmærksomhed på tailoring-aktiviteter, muliggør kommentarer og automatiske beskrivelser, tillad undersøgelse af en tailoring-fil, let administration og koordination og understøt en tailoring-kultur*. Denne afhandling forklarer disse forslag og beskriver forskningsprocessen. Denne omfattede litteraturundersøgelser og en efterfølgende udvikling og anvendelse af en prototypisk software (en tekstbehandlingsenhed og et groupware-søgeværktøj), der behandler aspekter af “collaborative tailoring”.

Afhandlingen består af to dele. I afhandlingens første del samles, fokuseres og uddybes de forskellige resultater og indfaldsvinkler i artiklerne i anden del (artikelsamlingen). Første del af afhandlingen kan læses som et bidrag i sig selv, hvori der ikke blot rejses metodiske spørgsmål, men hvor også selve processen med at udvikle forslag til “collaborative tailoring” beskrives og diskuteres.

De syv artikler i samlingen bidrager på forskellige måder til det overordnede forskningsspørgsmål. Den første og anden artikel belyser ”tailoring” i al almindelighed, præsenterer de første resultater af „collaborative tailoring” og fremhæver nødvendigheden af mere forskning i emnet. Den tredje, femte

og sjette artikel præsenterer de to cases: et tekstbehandlingssystem og et groupware-søgeværktøj. Den fjerde artikel præsenterer en metode til brugbarhedstest som et resultat af mit arbejde med tekstbehandlingscasen. Den syvende artikel præsenterer de forslag til “collaborative tailoring”, der er resultatet af tidligere arbejde. Et resumé af hver af de syv artikler i samlingen findes i introduktionen.

Table of Contents

SUMMARY.....	i
RESUMÉ.....	iii
TABLE OF CONTENTS.....	v
PREFACE & ACKNOWLEDGEMENTS.....	viii
REFERENCES TO PAPERS IN COLLECTION	ix
FIRST PART OF THESIS	
1. INTRODUCTION	1
1.1. Motivation & Research Objective	1
1.2. Research Approach.....	4
1.3. Structure of Thesis.....	7
2. FOUNDATIONS & RELATED WORK.....	10
2.1. Tailoring Software.....	11
2.2. Collaborative Aspects of Tailoring.....	18
2.3. Conclusions	26
3. TWO CASES	27
3.1. Word Processor.....	27
3.2. Groupware Search Tool.....	31
3.3. Review of the Cases.....	36
4. SUGGESTIONS FOR SUPPORTING COLLABORATIVE TAILORING	37
4.1. Continuing the Cases	38
4.2. Eight Suggestions	41
4.3. Discussion of Field Test	52
5. CONCLUSION	54
5.1. Results	55
5.2. Future Work.....	56
6. REFERENCES.....	57

SECOND PART OF THESIS

FIRST PAPER: FROM TAYLORISM TO TAILORABILITY	64
SECOND PAPER: HOW TO MAKE SOFTWARE SOFTER - DESIGNING TAILORABLE APPLICATIONS	74
THIRD PAPER: MORE THAN WORDS - COLLABORATIVE TAILORING OF A WORD PROCESSOR.....	100
FOURTH PAPER: CONSTRUCTIVE INTERACTION AND COLLABORATIVE WORK	128
FIFTH PAPER: DEVELOPING GROUPWARE WITH EVOLUTION AND PARTICIPATION - A CASE STUDY	140
SIXTH PAPER: TAILORING BY INTEGRATION OF COMPONENTS: THE CASE OF A DOCUMENT SEARCH TOOL.....	163
SEVENTH PAPER: COLLABORATIVE TAILORING – EIGHT SUGGESTIONS.....	199

Preface

Nowadays, most of the software is tailorable, i.e. it can be modified while it is in use in the field. This is fine considering the diversity, uncertainty and dynamism that software use in different organizations is subject to. Taking into account that in many groups people have similar tasks and that their computers are networked, the question arises, how collaborative tailoring can be supported adequately.

An answer is provided by this thesis which, of course, is subject to numerous limitations. Limited was the time to provide the answer, limited is my scientific background of and in Human-Computer Interaction and Computer Supported Cooperative Work, limited is my intellectual capacity, and reasonably large but still limited is the empirical and theoretical material on the issue. Therefore, there is still an unlimited abundance of aspects to understand and work to do, related to collaborative tailoring. However, as limited as the given answer may be, it is the first extensive treatment of collaborative tailoring. It recognizes and exceeds the existing literature of descriptions of how people tailor together, brings in my experience as action researcher and focuses on collaborative tailoring as a particular form of collaborative work.

Thus, the answer provided by this thesis is reasonable enough to be the basis for more refined questions and hence may become part of the slow process of evolution of knowledge about people working and using software collaboratively.

Acknowledgements

Many people have contributed directly or indirectly to this thesis.

I thank my supervisors Keld Bødker from Roskilde Universitetscenter and Peter Carstensen from the IT University of Copenhagen for their interest and willingness to work with me as an external Ph.D. student notwithstanding the organizational obstacles going along with this and for their valuable guidance and comments.

I am indebted to all members of the Research Group on HCI and CSCW (ProSEC) of the University of Bonn. ProSEC was a permanent source of inspiration and my scientific home for the last years. Particularly the discussions and work with my colleagues Oliver Stiernerling and Volker Wulf on many aspects of tailoring provided valuable insights.

Numerous friends contributed to the thesis as they let me forget the impenetrable thicket of collaborative tailoring once in a while and joining me in other interesting and enjoyable collaborative or colleisurative activities.

Most of all I thank my family for their patience and support.

Helge Kahler, Bonn 2001

References to Papers in Collection

The following papers are part of the collection of papers used for this thesis. Note, that where a contribution is written jointly, the first author is also the prime author. Their layout has been changed to adjust it to the first part of the thesis in order to improve the graphical appearance and readability of the thesis:

1. Kahler, Helge (1995): *From Taylorism to Tailorability*. In: Proceedings of HCI '95, Vol. 20B. Elsevier, Amsterdam. pp. 995-1000.
2. Stiernerling, Oliver; Kahler, Helge; Wulf, Volker (1997): *How to make software softer - designing tailorable applications*. In: Proceedings of DIS '97. pp. 365-376.
3. Kahler, Helge (2001): *More Than WORDs - Collaborative Tailoring of a Word Processor*. To appear in: Journal of Universal Computer Science (j.ucs), Vol. 7 (9) (September 2001).
4. Kahler, Helge (2000): *Constructive Interaction and Collaborative Work: Introducing a Method for Testing Collaborative Systems*. In: acm interactions, Vol. VII (3) (May/June 2000). pp. 27-34.
5. Kahler, Helge (1996): *Developing Groupware with Evolution and Participation - A Case Study*. In: Proceedings of PDC '96. Computer Professionals for Social Responsibility. pp. 173-182.
6. Kahler, Helge; Stiernerling, Oliver; Won, Markus; Wulf, Volker (2001): *Tailoring by Integration of Components: The Case of a Document Search Tool*. Resubmitted for publication at Transactions of CHI after first round of Transactions of CHI review process.
7. Kahler, Helge (2001): *Collaborative Tailoring - Eight Suggestions*. Technical Report 2001-01 of the International Institute for Socio-Informatics, Bonn.

First Part of Thesis

1. Introduction

This section provides information on my motivation and research objective for the work on this thesis, describes the research approach taken, and provides the structure of the thesis at hand including summaries of the papers of the collection part of the thesis.

1.1. Motivation & Research Objective

Since 1991 I have been working in the fields of Human-Computer Interaction (HCI) and Computer Supported Cooperative Work (CSCW), first as a student, later as a researcher. Over the years I participated in several projects, all of which were concerned with the issue of using computers to support professional work and involved application partners, i. e. organizations or groups which were observed how they worked or tried out software that was provided for them. The general questions relating to these application partners have always been *What does the work there really look like?* and *What can I derive for the design and introduction of computers to support office work?* Over the time I experienced many things that had been and still are described in literature: work life has many facets, you should not rely on organizational charts, the informal aspects of work often matter more than the formal aspects.

This continuous work made clear that one of the major challenges was to find adequate ways to deal with this heterogeneity and dynamism. One of the ways is to continue design in use (or *Design at Work*, see Greenbaum & Kyng 1991).

Most of the projects, I participated in, therefore involved user participation or evolutionary prototyping in order to understand the concrete user requirements and to be able to refine the systems to be used in a stepwise fashion. In this research setting it was just natural to consider tailorability, i. e. the possibility to modify the functionality of technology while the technology is in use in the field, as one option to cope with the diversity, uncertainty and dynamism of the application partners and to continue design in use. The papers in the collection part of the thesis (see subsection *Structure of Thesis* below for an overview over those papers) reflect this work: they document either long-term case studies dealing with particular

aspects of system use in selected organizations or papers on different aspects of tailorability based on experiences in concrete organizations or about proposing, implementing and testing certain aspects of tailorability for a system.

Four research projects shaped my understanding of designing in use and provided many insights without which this thesis had not been possible:

- GvS (Gestaltung vernetzter Systeme / design of networked systems, 1991 - 1993) focused on German design norms and guidelines for single user applications that were transferred and enhanced to include groupware aspects like suitability of information, moderability, visibility, controllability of interactional influence, and particularly group-oriented configurability (see Herrmann et al. 1996). The project included intensive cooperation with two application partners.
- POLITEAM (1994 - 1998) installed and maintained a groupware system at two sites at the German public administration. This involved user participation and an evolutionary process of software development and introduction (see Klöckner et al. 1995, Pipek & Wulf 1999).
- Virto (1996 - 1998) and InKoNetz (Integriertes Kooperationsmanagement in Netzwerkorganisationen / integrated cooperation management in network organisations, 1998 - 2000) worked on the design and introduction of groupware for the internal cooperation of virtual and networked organizations. This covered both theoretical and practical aspects (see Kahler & Rittenbruch 1999, Lemken et al. 2000, Rittenbruch et al. 1998).

Working in these projects, I learned how helpful it could be for individuals to tailor their systems to their needs. It also became obvious that tailoring collaboratively was sometimes necessary and always promised to increase efficiency and cohesion within a group. Obviously, using groupware applications (as the application partners sometimes did) or at least a computer network (as they always did) was a good prerequisite for distributing and sharing tailoring files as one form of collaborative tailoring which for many of the application partners seemed a useful thing to do. So there was a need –also described in the literature– as well as an option for improvements.

In the course of these projects a question evolved that then guided my further research: *How can collaborative tailoring be adequately supported?* The thesis at hand aims to answer this question by providing theoretical and empirical insights, as well as analytical and constructive results. The first part of the thesis and the papers of the collection in the second part of the thesis contribute to this in two ways: First, the thesis is *analytical* by providing aspects of collaborative tailoring in different settings. In the first part of the thesis this is done by two means: a comprehensive literature survey of tailoring in general, and particularly collaborative aspects of tailoring, presents the publicly available state of the art. The presentation of some of my own work adds to this by providing examples of implementations of aspects of collaborative tailoring in two cases. The collection of papers in the second part of the thesis provides more material regarding research on the work in different organizations, aspects of tailoring and the two cases (see the summaries of the papers and figure 1 below for more detail). This leads to the second research objective: the thesis aims at being *constructive* by deriving suggestions for collaborative tailoring from the preceding description with a particular focus on software tools and software features. These suggestions may then serve for researchers to refine and transfer to other areas, for software developers to have a guideline for implementing necessary functionality and provide adequate software structures, and for practitioners to be able to select and tailor generic software and to provide organizational structures that support collaborative tailoring.

This thesis is based on intensive literature study and collection of descriptions of related work, on my experience as action researcher, particularly with tailoring and collaborative tailoring, and the identification and implementation of identified relevant features in software and their evaluation in laboratory settings and a field test. Thus, the thesis is the first extensive treatment of collaborative tailoring. It recognizes and exceeds the existing body of work of descriptions of how people tailor together and how some features of collaborative tailoring are implemented on the background of a particular technology or system. At the same time it focuses the abundant literature about collaborative work on the issue of collaborative tailoring.

Tailorability can be treated from different perspectives. One of the requirements to be found in much of the literature on tailorability is that it should not be relating only to the surface, but particularly to functionality

deeply embedded in the application (see e.g. the *deep customization* proposed by Bentley & Dourish 1995). This in turn led to research in the field of software engineering interested in software architectures for tailorability (see e.g. JCSCW 2000 Vol. 9 (1), Special Issue on Tailorable Systems and Cooperative Work for several papers dealing with this). A second, product oriented, perspective on tailorability is concerned with how tailorability is or might be realized in (commercial) software products. As a third alternative, tailorability can be treated from a theoretical perspective. The perspectives overlap and influence each other. This thesis centers in the area of theory: factors relevant for collaborative tailoring are identified, ordered, and generalized. The *suggestions for supporting collaborative tailoring* are the result of this process. However, the other perspectives are involved. One of the most promising software engineering approaches to tailorability are *component-based architectures*. Such an approach has also been used in the groupware search tool case described below. Also, the proposed suggestions are partly based on research of (enhanced) commercial software (word processor and groupware case) and are meant to influence not only a theoretical debate but also non-commercial and commercial software development.

1.2. Research Approach

The work described here has been conducted with a background of Human-Computer Interaction (HCI) and Computer Supported Cooperative Work (CSCW). The references used for this thesis show the influence of both disciplines. Most of the work on tailorability of single user applications can be found in the field of HCI (e.g. Trigg et al. 1987, Mackay 1991), whereas most of the general aspects of collaboration are covered by the CSCW literature. Interestingly enough, the literature illuminating the collaborative aspects of tailoring derives from both research fields (e.g. Gantt & Nardi 1992 for HCI; Bentley & Dourish 1995 for CSCW).

The chosen research approach is - or rather the approaches are - based on knowledge of, and experience in these two research fields. Since collaborative work is a complex matter, qualitative research with its particular strength to explain what goes on in a group or organization, has gained acceptance during the last years (see Avison et al. 1999). Several approaches have been involved: For each part of the research the approach which seemed most adequate was chosen. The underlying assumption for

the work described here is that human behavior is always bound to the context in which it occurs and that social reality cannot be reduced to variables in the same ways as physical reality. In this sense, all my research is qualitative. The research approaches used are described and discussed below.

Action Research

Action research is valuable to enhance the understanding of a complex human process rather than being a universal prescriptive truth. It is an interventionist approach to acquisition of scientific knowledge linking theory and practice and involves problem diagnosis, action taking, evaluation and learning. The collaboration of research scientists with practitioners is a key aspect (Baskerville & Wood-Harper 1996). Action research is suitable for theory building and for finding out something about organizations or groups (Galliers 1991).

Like most of the other projects I was involved in, the POLiTEAM project was engaged in an action research oriented approach, and subsequently were the endeavors for an adequate search tool at the application partners' organizations (see fifth paper and sixth paper of collection). Over a period of several years I studied the organizations and together with them I worked on changing and shaping their use of information technology.

Survey

A survey is a snapshot of practices, situations or views, usually executed by questionnaires or interviews and is suitable for theory building and for finding out something about organizations or groups (Galliers 1991).

Structured interviews have been used as a means for data collection at several points in different areas of the work described here (see third paper, fifth paper, and sixth paper). They either served to support an initial understanding of relevant aspects (e.g. experiences in collaborative tailoring) or to contribute to an evaluation of a prototype.

Laboratory Test & Field Test

Laboratory experiments or tests aim at the identification of relationships between variables in a controlled environment. Field experiments extend laboratory experiments to a "real world" area out

of the laboratory in order to receive results from a less artificial environment than a laboratory. Both laboratory and field experiments are suitable for technology and theory testing (Galliers 1991).

For the evaluation of prototypes of the word processor extension and the groupware search tool and its included tailoring environment, as well as the underlying assumptions on collaborative tailoring, tests in the laboratory (see fourth paper) and in the field (see sixth paper and section *Suggestions for Supporting Collaborative Tailoring*) have been used. The laboratory test of one version of the word processor extension was performed with a small number of participants and the new method *constructive interaction for testing collaborative systems - (CITeCS)* developed in the context of the work described here. Another version of the word processor extension will be tested in a group of about 5 to 15 people for several weeks. The last version of the groupware search tool tailoring environment has been applied in the organization where it had been developed in the context of action research. Since its functionality is very advanced and its implementation was equally inspired by the needs of the organization and research interest, the test can be considered either to be a part of action research or a field test.

One reason for taking different research approaches was to provide an adequate approach for each of the different research activities. Action research helped to obtain a deeper understanding of the organizations I worked with particularly over time. However, gaining insight into a specific problem, such as collaborative tailoring, took a long time due to the holistic approach and process.

The interviews provided an overview over a particular aspect at one time in the interviewees' work life. Interviews with people from different organizations helped to get a broad view on different individual and organizational tailoring habits in the case of the word processor and provided feedback on the subjective usability and utility of the search tool tailoring environment, respectively. The data of the interviews were less rich than those of the action research.

The laboratory test was rather focused on the usability of the word processor extension and provided results for this aspects. However, for obvious reasons it lacked a real organizational embedment; arguably, it would have been good to increase the number of tested persons in order to increase its

reliability. The field test in the search tool case concluded a set of activities in that one organization. This resulted in the methodical issue, though, that the participating persons knew much of the prototype and its development before the test started. This could have hardly been avoided as only the fact, that previous action research including workshops in this organization had created trust on their side and the necessary knowledge on my side, allowed the introduction of such a complex tool into this organization.

Each of the approaches contributed its advantages to the research. The usage of several different approaches in different settings for different, but related aspects of collaborative tailoring, thus provided a cross-approach tri- (or multi-)angulation to provide additional support to the generalizability of the results adding to the generalizability that is provided by every single research effort. While it may not be reasonable to attribute particular methodological results to particular research approaches, I may say that the action research and the survey undertaken account for much of the results' ecological validity while the reliability is mainly covered by the laboratory and field tests.

1.3. Structure of Thesis

The thesis consists of two parts. In the first part the various results and insights of the second part (*collection*) are gathered, focused, enhanced, and related to each other. The first part of the thesis can be read as a contribution in its own right, where not only methodological issues are raised but also the process of generating and field testing suggestions for collaborative tailoring is described and discussed.

The seven papers of the collection contribute to the overall research question in different ways. A summary of the papers and their position in the matrix of approaches and results, similar to the one suggested by Sørensen (1994) (see figure 1), are provided in the following. Note that I do not intend to claim that there is a continuum between analytical and constructive result or between theoretical and empirical approach. Rather, figure 1 serves to depict how the approaches and results are represented in the respective papers.

The order of the papers in the collection part of the thesis is determined by the structure of the first part of the thesis rather than chronological. The first and second paper illuminate tailoring in general, present first results on

collaborative tailoring and highlight the necessity for more research about collaborative tailoring. The third and fifth paper present the two cases: word processor and groupware search tool. The fourth paper demonstrates a method for usability testing resulting from my work on the word processor case. The sixth paper presents among others the results of a field test of a search tool and its tailoring environment which had aspects of these suggestions implemented. The seventh paper documents the suggestions for collaborative tailoring derived from the previous work. All papers present work dealing with different aspects of my research question of how collaborative tailoring can be adequately supported.

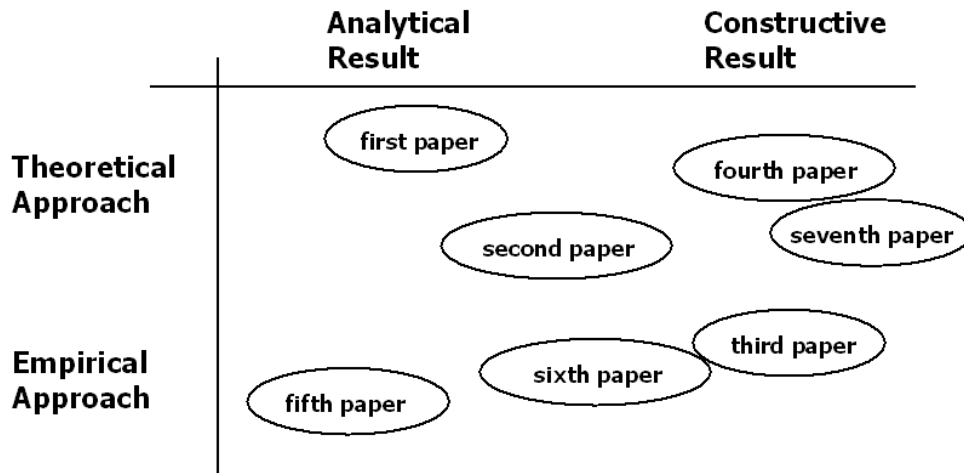


Figure 1: Position of papers in research matrix

1.3.1. From Taylorism to Tailorability (first paper)

The first paper relates tailorability to the division of labor proposed by F.W. Taylor. It is argued that collaborative forms of tailoring have merits particularly for post-tayloristic organizations and several examples are provided. The paper concludes with the request for more research on collaborative tailoring. It is the basis and starting point of my work presented in this thesis.

1.3.2. How to make software softer - designing tailorable applications (second paper)

The second paper relates tailoring in a groupware setting to positive experiences with participative and evolutionary software development. A broad approach to the requirement analysis for tailorable applications is advocated and two examples from our own work are provided. This paper contains first results of my own field work related to a broad approach to come up with design suggestions for tailorable applications which built the background for my further work.

1.3.3. More Than WORDs - Collaborative Tailoring of a Word Processor (third paper)

In the third paper the word processor case is presented. Based on previous knowledge about collaborative tailoring, it is described how an extension of a word processor supporting collaborative tailoring has been implemented and evaluated. This case contributed to the subsequent suggestions for collaborative tailoring, particularly through the treatment of issues of sending and receiving tailoring files. Also, the case served as one concretization of the more abstract treatment of collaborative tailoring in the first and second paper.

1.3.4. Constructive Interaction and Collaborative Work: Introducing a Method for Testing Collaborative Systems (fourth paper)

In the fourth paper a new method for testing collaborative systems (constructive interaction for testing collaborative systems - CITeCS) is introduced. This method is a side result of my work on the word processor case and served for the laboratory test of the word processor. Thus, it is an indirect contribution to the suggestions for collaborative tailoring and depicts this particular form of laboratory test and its contribution to my research question.

1.3.5. Developing Groupware with Evolution and Participation - A Case Study (fifth paper)

The fifth paper describes the basic setting of the groupware search tool case and the first steps of the development of a prototype which is not yet

tailorable. The evolutionary and participatory action research approach becomes clear and the basis is laid for further development and a subsequent field test of the search tool and realized aspects of the suggestions for collaborative tailoring (see sixth paper).

1.3.6. Tailoring by Integration of Components: The Case of a Document Search Tool (sixth paper)

In the sixth paper the steps of the development and field test of the collaboratively tailorable component-based groupware search tool are presented. For this thesis, particularly the field test of aspects of the suggestions dealt with in the seventh paper as realized in the groupware search tool tailoring environment is relevant. The field test shows the general usefulness of the implemented suggestions and hints at further need for research, particularly to handle the complexity of such a powerful tool.

1.3.7. Collaborative Tailoring - Eight Suggestions (seventh paper)

The seventh paper presents the eight suggestions for collaborative tailoring derived from my work described in the first five papers. The background for each suggestion as well as examples and the relation between technical and socio-organizational aspects are provided. Whereas the other papers focused on describing a process and a subsequent result, the seventh is primarily a presentation of results from a broad range of previous work. Thus, it contains one possible answer to my research question of how collaborative tailoring can be supported adequately.

2. Foundations & Related Work

In this section the main relevant literature for approaching collaborative aspects of tailoring is presented. A large body of work has been devoted to theoretical and empirical work on how tailoring and particularly collaborative tailoring of software could be supported technically, why it should be supported and how software has been tailored by users. Some authors also described systems which they implemented and which have sometimes been evaluated. The presented literature is restricted to contributions explicitly discussing tailorability as a major issue. Since nowadays much software is tailorable in one way or another, this restriction was necessary to stay focused. However, it would certainly be worthwhile to

take a closer look at tailorable software and literature about it, where tailorability is not the main focus of the paper or the most outstanding feature of the software.

2.1. Tailoring Software

Modern software usually comes with a cornucopia of features, that permits many forms of file creation or modification. However, this abundance very often makes it difficult for users to find out how to perform a particular task with the software or how to do so efficiently. *Tailoring* is the technical and human art of modifying the functionality of technology while the technology is in use in the field. Consequently, the feature of a software that states that the software can be tailored is called *tailorability*. It is widely agreed that tailorability is one of the major future challenges in the design of interactive systems (Bentley & Dourish 1995, JCSCW 2000 Vol. 9 (1), Special Issue on Tailorable Systems and Cooperative Work).

2.1.1. Reasons for Tailoring

The main and overall reason why software should be tailorable and needs to be tailored is the *complexity* of the setting where it is used and of the task it is used for. Woods (1988) (as quoted in Carstensen 1996, p. 52) distinguishes four aspects of complexity: the *dynamism of the world*, a *large number of interacting parts*, *uncertainty* e.g. about available data, their inference and future states, and the existence of *risk*, where the possible outcome of choices may cause high costs. While Woods provides a general overview over complexity, Trigg (1992) focuses explicitly on tailorability. Trigg argues in favor of tailorability and considers it to be beneficial and basically less risky if a choice is made by a human than automatically by the software. He provides three main reasons why systems should be tailorable.

- The *diversity* along several dimensions like persons, tasks or objects of tailoring must be taken into account when selling a generic software that is supposed to fit different settings. A lawyer in Saudi Arabia probably uses a word processor differently than a researcher in Europe does. This even holds true for custom-made software: people may work in the same office with different tasks or even only different usage patterns of mouse and keyboard. And also one single person may want to perform a single task differently at different times or related to a different context.

- The *dynamism* (called *fluidity* by Trigg) of individual and organizational work corresponding to the changing nature of work over time requires software to also change over time. The structures of work organization and collaboration may vary considerably in relatively short time periods.
- The *uncertainty and ambiguity* about the exact work practices and procedures, even in the perception of the workers, makes it necessary to leave room for alternative ways of performing tasks. Trigg (1992) reports a case, where a person in an interview about her work practice revealed uncertainties of her own view on her work situation, e.g. how reasonable it was to archive certain documents.

Based on similar observations Oppermann & Reiterer (1992) derive the need for tailorability from different perspectives:

- From a *software technological* perspective tailorability bridges the gap between development and use. They argue that off-the-shelf software is in a dilemma: It is supposed to fit for many users, many tasks and for a long time span. To do so, it is shipped with settings for ascertained or assumed *average* requirements. Thus, it constitutes a compromise between the requirements of all users with all tasks at all times, resulting in an unmanageable software, and a particular user with a particular task at a particular time, resulting in an almost useless software. Oppermann & Reiterer argue that this constitutes a need to extend the development phase into the use phase, in order to enable developers or users to change the settings for the average use situation to meet their own needs.
- From a *work science* perspective, it is argued that the respective literature based on ISO 9241 Part 10 (ISO 9241: Ergonomic requirements for office work with visual display terminals (VDTs), Part 10: Dialogue principles) leads to tailorability, particularly the dialogue principle *suitability for individualization*, but also the principles *conformity with user expectations*, where tailorability supports that the conformity can be established by the users themselves, and the principle *controllability*, where tailorability allows users to reach more control over the selection and order of program steps and the kind and scope of in- and output.

- Taking an *organizational* perspective, Oppermann & Reiterer claim that organizations are highly complex and dynamic social networks that require a software system to be able to be modified to this dynamism.

Grudin (1991) hints at the fact that software is almost never programmed to be used by one person at one time, but by many users at many times, who are often not personally known to the programmers, or who perform a task unknown to the programmers.

Haaks (1992) distinguishes different dimensions of tailoring, including initiator and actor, object, aim, time, and scope of validity. The *initiator and actor* can be the system or a user. The *object* of tailoring depends on the taken perspective and can range from setting default values, via limiting the available functionality in order to ease the learning of the system, to modification and enhancement of functionality. The *time* when a system is tailored may also differ. Tailoring can take place before the first use of the system, between different phases of use and during use. Oberquelle (1994) does not consider the first case to be tailoring, but configuring, since it is not a reaction to local needs during use. Haaks (1992) concedes that only tailoring a system before the first use is never enough since it does not take into account the dynamism of use. He describes the *scope of validity* as depending on the concerned group of persons (a single user, a group of users or all users), the time span (a session or a phase of usage) and the affected aspects of the software. Wulf et al. (1999) claim that the definition of the scope of validity is particularly important for tailoring groupware applications, and define system behavior for different scopes of validity by creating and ranking tailoring statements.

Note that the literature about tailorability often conveys the impression that tailoring is an explicit process. While this is certainly often true, tailoring may also be performed implicitly, e.g. as part of the ordinary work of a person (see below for the distinction of using and tailoring a software).

These reasons and dimensions for tailoring require that for a concrete tailoring activity the work practice of the group to be supported must be known. Kjær & Madsen (1994) applied participatory techniques in analyzing the requirements for flexibility of a picture archive and communication system in a hospital. They support the notion of dynamic work and write about flexibility that is part of a group's work practice (p. 22):

“[...] flexibility concerns not the regular procedures and standard way of doing things but the unexpected, unprecedented, the exceptional cases, situations and events which are only experienced by the people who do the day to day work.”

On one hand this underlines the importance of tailorable systems allowing users to transfer the flexibility of their work to the software they use. On the other hand it supports the argument that people who tailor systems for or together with others need to know a good deal about the others' work. The importance of local experts is highlighted in the subsection *Collaborative Aspects of Tailoring*.

2.1.2. Definitions

There are different terms, concepts and taxonomies connected with the idea of users modifying software during use, such as *individualization*, *personalization*, *adaptation*, *customization*, *end-user modification*, and *tailoring*. More than 20 years ago the EMACS editor provided mechanisms for extension by the user while it was running (Stallman 1981, p. 149):

“Many minor extensions can be done without any programming. These are called customizations, and are very useful by themselves.”

Some years later, a more thorough distinction is provided by Trigg et al. (1987, p. 723) who described Xerox PARC's information structuring system NoteCards:

- *“a system is flexible if it provides generic objects and behaviors that can be interpreted and used differently by different users for different tasks*
- *a system is parameterized if it offers a range of alternative behaviors a user can choose among*
- *a system is integratable if it can be interfaced to and integrated with other facilities within its environment as well as connected to remote facilities*
- *a system is tailorable if it allows users to change the system itself, say, by building accelerators, specializing behavior, or adding functionality”*

The authors stress the importance of tailorability, since it is the one of these features that allows users to change the system behavior in ways unanticipated by the system's designers.

Beginning in the late 1980ies, Oppermann and his colleagues at the GMD worked on the topic of adaptation of software. Based on the notion of individualization demanded by ISO 9241 Part 10 (see above), they studied adaptive systems (cf. Oppermann 1994 for a comprehensive report). They distinguish between *adaptivity*, being system-initiated individualization, and *adaptability*, being user-initiated individualization. Kühme et al. (1992) classify systems by employing the four steps *initiative*, *proposal*, *decision* and *execution* of a system change, where in purely adaptive system all of the steps are taken by the system and in an adaptable system all are taken by a human. This allows the description of hybrid systems, where e.g. the initiative and decision are performed by the user and the proposal and execution are performed by the system ("computer-aided adaptation"). Oppermann (1989) considers adaptivity and adaptability to be the two sides of individualization which, together with *variety* (i.e. the possibility to choose from a number of options) constitute *flexibility*.

Fischer & Girgensohn (1990, p. 184) provide a taxonomy of *end-user modifiability* where

"the changes supported by a modifiable system include the following [...]:

- setting parameters (e.g., with the help of property sheets),*
- adding functionality to existing objects,*
- creating new objects by modifying existing objects, and*
- defining new objects from scratch"*

They stress (p. 184) that

"end-user modifiability makes systems adaptable, in contrast to adaptive systems [...] which change themselves based on the user's behavior."

Mørch (1995a) defines *tailoring* as being the adaptation of generic software applications to the specific work routines of a user organization. Based on a literature survey he identifies three levels of tailoring:

- *Customization* means selecting among a set of pre-defined configuration options by direct interaction or setting parameters;
- *Integration* can be *hard integration*, where a component is attached physically to the application, or *soft integration*, where a component is integrated by means of a macro, script, or agent;
- *Extension* means adding new code to the application.

Similarly, Henderson & Kyng (1991) distinguish the three levels choosing between alternative anticipated behaviors, constructing new behaviors from existing pieces, and altering the artifact. They consider these to be activities that the tailors must do related to the above mentioned four properties of systems by Trigg et al. (1987).

Henderson (1997, p. 1) defines

“Tailoring is the technical and human art of modifying the functionality of technology while the technology is in use in the field.”

This definition is adequately broad and includes both the relevant technical and socio-organizational aspects. It also stresses the importance of really using a technology at some place and in particular work settings in order to be able to know how it should be tailored there. Therefore, this definition is also used for the thesis in hand.

2.1.3. Tailoring, Using, and Developing

Tailoring software can be distinguished from use and development, although it bears similarities with both. On one hand, it is a way to continue design in use to account for unanticipated needs; on the other hand, it extends use by providing means to make it effective and efficient. Henderson & Kyng (1991) argue with the relative stability of an application, claiming that people tailor when they change stable aspects of an artifact. However, they also admit that the distinction may be difficult: Changing the font of a document can be considered to be use or tailoring. They also introduce the notions of *subject matter* vs. *tool* of work and claim that changing the subject matter is use, while changing the tool is tailoring. Again, the distinction is not always clear, since one person’s subject matter is another person’s tool: For a person using an application programmed in C++, this application is a tool, whereas for its programmer it is the subject

matter, and the C++ compiler is the tool (and for the compiler builders it is the subject matter). Finally, if the effect of a modification is immediate only, the action can be considered to be use.

2.1.4. *Tailoring in Action*

When tailorability was still a rare feature for software, several systems have been described where different forms of tailorability play an important role. Note that the core of explicit discussion of tailoring and the high time for implementations based on the idea of tailoring lies in the early 1990ies. Later on, tailorability was considered to be a common feature for software, although many interesting concepts had never been implemented or tested on a broad scale and tailoring possibilities often consisted in mere option-picking or technical parameter-setting being of dubious value to users.

With SHARE Greenberg (1991) proposed a layered architecture for a conference tool with the possibility to use one of a set of personizable floor control policies, i. e. rules for deciding who's turn it is next to speak or write in a computer-mediated conference. Such a rule could state that every person, who wants to say something, can just take the turn by interrupting the speaking person, or that the speaking person has to explicitly stop speaking before anybody else can start. Also, there could be a chairperson, who picks the next speaker from the group raising their hands. The decision for a special policy is made before every single session. Malone et al. (1992) describe a system that allows end users to tailor software on a level closer to system development than just setting parameters. Their OVAL System is a radically tailorable tool for cooperative work, where *radically tailorable* means, that the tool is meant to enable end users to create different applications by modifying a working system. This is done by combining and modifying objects, views, agents, and links, which provide an elementary tailoring language. While, in fact, the idea of end users designing the application that suits them best, is intriguing, the question remains how many users were able to handle the more advanced features of this complex system.

Mackay (1991) provided valuable insights to actual tailoring. She studied the tailoring behavior of 51 users of a Unix software environment over a period of four months. Four main reasons that lead to tailoring have been identified: external events like job changes or office moves, social pressure like contact to colleagues who suggest changes, software changes like

breakdowns or upgrades (the latter often retrofitting new software to behave like the old version), and internal factors like spare time or running across a previously unknown feature. The topmost barriers for the persons she asked were the individual factor *lack of time* (cited by 63% of the users) and the technological factor that the software was *too hard to modify* (33%). MacLean et al. (1990) present the *Buttons* system, where the buttons are tailorable Lisp screen objects that allow users to carry out an action. They claim that users even with little or no programming experience can modify various aspects of the buttons, and that the architecture is powerful enough to allow users with programming skills to create new buttons for novel applications. They identify nine tailoring techniques in Buttons and claim that they are in an ordered relation so that they build steps to climb the “*slope of tailorability*” (p. 181). Providing these relatively small steps to the top of the “*tailorability mountain*” (p. 175), the authors claim that there is only a little barrier to learn a new, more advanced tailorability feature.

2.2. Collaborative Aspects of Tailoring

Since more and more work with a computer is done in groups where people work on similar or the same tasks and files, tailoring very often has collaborative aspects to it. Collaboration here is defined in a rather broad sense. Several arguments have been put forward to distinguish collaboration from cooperation. Dillenbourg et al. (1995) grant that both terms are often used interchangeably and that there is some disagreement amongst the authors themselves. They then continue to define that in cooperation, the task is split hierarchically into independent subtasks; whereas in collaboration, cognitive processes may be heterarchically divided into intertwined layers. Grudin (1994) associates cooperation with small groups, where relatively little friction or discord among users is anticipated and collaboration with larger organizational systems, where conflicting goals play a major role. Bannon & Schmidt (1991) note that distinctions between such terms as cooperative work, collaborative work, collective work, and group work are not well established in the CSCW community (which has not changed much since then in that respect). After a literature overview they conclude that collective work, collaborative work, coordination and articulation work designate different types or facets of cooperative work. Collaborative work, in their view, stresses a particular complying spirit among the cooperators.

Since this thesis focuses on rather small groups and collaboration in tailoring is generally done voluntarily, I assume that this complying spirit does exist and am going to use the term *collaboration* for common tailoring activities considered here, where collaboration is a sub-category of *cooperation* so that the following statements about cooperation also hold for collaboration.

Clarke (1996) also provides an overview over cooperation based on literature study. He distinguishes three advantages of cooperation, namely (p. 59)

- “1. *Efficiency, where cooperation minimises the effort required to achieve the goal.*
2. *Possibility, where the partners can achieve goals not possible for one person to achieve, and*
3. *Synergy, where the cooperating partners can together achieve a different order of result from that achievable separately.*”

He also distinguishes the three types of cooperation: *full cooperation*, like in the tightly knit cooperative working of a small group like a surgical team, *intermediate cooperation*, like in various manufacturing activities, and *loose cooperation*, where people share goals and rewards like shareholders of the same company wanting to make money.

Nielsen & Carstensen (1998) stress the fact that cooperation among people is constituted by interdependence between tasks which creates an interdependence of actors. Referring to Schmidt’s (1994) work on *Modes and Mechanisms of Interaction in Cooperative Work*, they provide an example from maritime navigation where they analyze a scenario of approaching a harbor where the captain and several other persons interact. In this situation, the success of bringing the vessel safely into the harbor vitally depends on the actors’ cooperation. They concede, however, that it is necessary to distinguish between actual and potential actor interdependencies.

In the literature on tailorability collaborative aspects have not played a major role so far, but have been mentioned at several occasions. Already Stallman (1981) reports that users not only think of small changes and try them, but also pass them over to other users. Mackay (1990) researched how people actively shared their tailoring files with each other. The study was

conducted at two research sites. At the first site, 18 people using Information Lens (Malone et al. 1988) to tailor the management of their emails were observed over a period of three or more months. This included several interviews per participant and the collection of automatically gathered data. Mackay reports that several people shared Information Lens rules (i.e. text files containing information about the filtering of mails), including two manager-secretary teams who used the rules to support a standard form of communication. At the second site, a group of 51 users on a common project sharing Unix tailoring files were observed over a period of four months. The data gathered stem from one or more interviews per user and also included copies of their tailoring files. More than three-quarters of the participants received tailoring files from others since they had joined the project. The following methods to obtain or give tailoring files or ideas were identified (p. 213):

- *“Someone helps you to get set up.*
- *You ask someone to help you get set up.*
- *You get the standard system file and use it.*
- *You have a problem and ask someone for help.*
- *New ideas are posted electronically in a common area and you look.*
- *Someone has a new idea and tells you about it.*
- *Someone tells you to look in the common area.*
- *You have a symbolic link to someone else’s file which is automatically updated.*
- *You walk by and see someone else’s screen and ask how something was done.*
- *You watch someone performing some task, notice a useful technique, and ask, how it’s done.*
- *You help a newcomer get setup with a version of your files.*
- *You post an idea in the common area.*
- *You tell your friends about a new idea.”*

Depending on the job category (e.g. Manager, Secretary or Application Programmer) the different groups borrow and lend files with different intensity and have a different percentage (0% to 38%) of *translators*. To

Mackay these are persons who actively share their files and talk to the recipients of the files. She concludes both cases by criticizing that staff members are often not rewarded for sharing tailoring files and requests that tailorable software should provide the ability to browse through others' useful ideas and that it should include better mechanisms for sharing customizations which then may serve to establish technical or procedural standard patterns.

The role of a local expert was also highlighted by Gantt & Nardi (1992) who describe what they call *patterns of cooperation among CAD users*. They studied the use of a Computer Aided Design (CAD) system by conducting in-depth interviews with 24 informants and collecting and analyzing the informants' CAD artifacts. They distinguish between *local developers* who write macros, programs and scripts and help end users in tailoring on one hand, and on the other hand *gardeners* as a sub-group of local developers. With gardeners, the informal position of a local developer has evolved into a formal or semi-formal position. They are responsible for writing and disseminating standard macros and programs at the corporate and department level. Usually, a gardener has both domain and computer knowledge and often starts from the domain side and then acquires the necessary computer expertise. Gantt & Nardi support the contention that the activities of local experts should be recognized and promoted since a local expert, and particularly a gardener, can save the organization's time and money by offering valuable resources, like macros and programs to the entire group. They admit, however, that it may be difficult to find a person with the right combination of technical and social skills.

Nardi & Miller (1991) report that spreadsheets offer strong support for cooperative development of applications. They present results from an in-depth-study based on data of 350 pages of transcriptions from interviews with 11 users of several spreadsheet products. Nardi & Miller conclude that spreadsheet co-development is the rule rather than the exception and that spreadsheets support the sharing of both programming and domain expertise. In their study they describe, how spreadsheet users (p. 163)

- *“share programming expertise through exchange of code;*
- *transfer domain knowledge via spreadsheet templates and the direct editing of spreadsheets;*
- *debug spreadsheets cooperatively;*

- *use spreadsheets for cooperative work in meetings and other group settings; and*
- *train each other in new spreadsheet techniques.”*

They identify the three kinds of users: *non-programmers*, *local developers*, and *programmers*. The first group is responsible for most of the development of a spreadsheet, and the second and third group contribute code to the spreadsheets of less experienced users. Local developers are technically less experienced than programmers are but serve as consultants for non-programmers in their work environment. For spreadsheet applications it can be argued that using and tailoring them are closer together than for many other applications, since their usage - in the sense that you just put in numbers and calculate something - implies the prior work of defining code behind the spreadsheet's cells which is responsible for the calculation. This is usually done by persons who have domain knowledge and, as Nardi & Miller note, usually done cooperatively. Considering the fact that more and more off-the-shelf software needs tailoring and offers mechanisms for it, the presented results encourage the tighter integration of using and tailoring.

Trigg & Bødker (1994) found an emerging systematization of collaborative tailoring efforts in a government agency. In their study, they examined the tailoring of word processors performed by four persons in a Danish administration over a year. After this, they conducted hour-long interviews. The four protagonists work on the borders between technology development and everyday work, two of them being officially recognized local developers whose tailoring work is part of their job description. The third one is a system supporter and the fourth person is the least technologically inclined of them. Tailoring at their organization mainly means customizing the word processors button panels, macros, and standard forms. Trigg & Bødker explicitly distinguish tailoring from programming, since the latter moves from analysis to design and then to realization, while the former basically consists of trial and error where the starting point often is a personal solution that may become more stable and then used by several people after a constructive process of small improvements. They observed that tailoring is often a collaborative process where the idea and the basic work is performed by the local developers who then pass on their partial solution to the programmer for improvement. Also, the learning process of tailors has distinctly collaborative character, as they ask each other and

consider themselves to be on a learning staircase trying to move upwards. Over the time, the sharing of tailoring files had evolved from an opportunistic spreading where someone heard about tailoring done by a colleague and copied their tailoring files to a more systematic activity: ideas are conveyed to the local developers or the programmer who then implement them. The new tailoring files are downloaded when a computer is rebooted (usually each morning). In particular cases, the workers are notified about how they are supposed to use the new functionality. They are also free to ignore the tailoring files. While it is often argued that tailoring leads to an unmanageable abundance of individualized solutions, several aspects imply that tailoring in this organization does rather have a standardizing effect. Standards of particular text blocks and of macros and button panels that reflect the work practice can be developed and widely used because the organization explicitly supports individual and collaborative tailoring and the distribution of tailored files.

Wasserschaff & Bentley (1997) describe how they supported collaboration through tailoring by enhancing the BSCW Shared Workspace system, which is an extension to a standard web server providing basic facilities for collaborative work. It includes information sharing, document management, and event logging and notification. They designed multi-user interfaces for the BSCW system, which allow users to take a certain view on the data in the shared workspace. These *Tviews* can be added to the shared workspace as objects in their own right, so others may take them, use them, and modify them in the same way as documents and folders. However, there are no evaluation data to support the notion of Wasserschaff & Bentley, that *Tviews* can bridge the gulf that often separates the tailoring of surface and presentation features from the deeper aspects of system behavior, nor is there data on actual collaborative tailoring performed around *Tviews*.

In their Buttons system MacLean et al. (1990) explicitly supported the sending of tailored files via email. They observed that via this opportunity, small improvements easily diffused throughout the user community and that a high amount of tailoring could be done by “*begging, stealing or borrowing*” (p. 178) appropriate buttons. Moreover, users often took buttons they had obtained from others as a basis to do some tailoring of their own by adding or changing something. MacLean et al. propose that the two possibilities to make systems more tailorable for workers are to make the tailoring mechanisms accessible and to make tailoring a community effort. The users in their study in the beginning had mixed feelings towards the

buttons and perceived them as unfamiliar and messing up the screen. However, after a while they got used to the buttons and did not only share buttons with others but also over time appropriated buttons and started to perceive them as personal and positive. The notion of the importance of a community of people who tailor is supported by Carter & Henderson (1990). Based on their experiences with the Buttons system they claim that a *Tailoring Culture* is essential to the effective use of a tailorable technology. Such a tailoring culture grows as tailoring becomes part of users' everyday work and makes them experience the technology as being under their control. Carter & Henderson conclude that (p. 113)

“tailorability is a relationship to rather than a property of technology. Tailorability addresses how technology fits into an organisation and how groups and individuals make use of it.”

As shown by the aforementioned examples, collaborative tailoring does not only occur among groupware users, but also in groups of users using the same software and thus being able to profit from the fact that this software is tailorable and that tailoring files may be exchangeable. Particularly the fact that more and more computers are connected to a local or wide area network creates the infrastructure to exchange tailoring files even of single user applications easily through the network. Therefore, the boundaries between collaborative tailoring of a single user software and a groupware become fuzzy.

Another aspect of collaboration is highlighted by Robinson (1993). He introduces the notion of *common artifacts* that are used by several people and may serve to provide awareness and an overview over the work process that otherwise would not be available. In Robinson's view, these common artifacts can provide useful hints for the design of systems to support cooperative work. In this way, tailoring files may serve as common artifacts for groups of people to learn about each others work and at the same time being a means to support forms of work that were not anticipated in the original system design.

Oberquelle (1994) proposes a classification of groupware tailoring distinguishing tailoring actors, who can be individuals or a group, from persons affected by a tailoring activity, who can again be individuals or a group (see figure 2). This can also be used to classify collaborative tailoring.

Different aspects and intensities of collaborative tailoring of a single user software and of groupware fit in the resulting four categories:

- *individualization*: individuals tailor for themselves and are the only ones affected by the tailoring activity – e.g. individual keyboard shortcuts or the window layout of an individual email client;
- *tailoring effective for group*: individuals can tailor for a whole group who then agree or are obliged to use the tailoring files - e.g. a system administrator or expert user provides a letterhead to be used by the group;
- *individualization supported by group*: a group can tailor synchronously or asynchronously for its members to use and change the tailoring file - e.g. several persons work on collection of macros that individuals can use;
- *group tailoring*: a group can tailor synchronously or asynchronously and its members agree or are obliged to use the tailoring files - e.g. several persons work on the introduction of semi-structured email templates valid for the whole group.

		Actors	
		Individuals	Group
Persons Affected	Individuals	Individualization	Individualization supported by group
	Group	Tailoring effective for group	Group tailoring

Figure 2: Classification of collaborative tailoring following Oberquelle (1994)

Collaboration certainly takes place where the whole group is actively tailoring. A weaker form of collaboration takes place where an individual, often a local expert, tailors for a group. Certainly, tailoring is more efficient in this case than if all individuals did it on their own. Also, the tailor and the group share the common goal of improving a system to meet their needs. Usually, as has been described in several of the studies mentioned above, there is also a form of interdependence involved. On one hand, the group depends on the tailor to supply something that meets their needs, and the

tailor needs to know the domain and the requirements from the group. On the other hand, since tailoring is often a constructive and cyclic process, the tailor usually depends on the group's feedback to improve the tailoring file. The border for individual actors tailoring for themselves and tailoring for the group is often blurry. Even if a person only tailors for herself or himself someone else might notice a difference to her or his own system and ask how the tailoring had been done and if they can have the tailoring file. Also, an individual might tailor something for her or himself and then think that it might be useful for two colleagues and send it to them or even for all colleagues and put it into a shared workspace. This, in turn, may lead to a second person changing the tailoring file and then again sharing it with others, which could lead to numerous versions and a discussion about them. Thus, even if originally there may have been no intention to interact or collaborate, the potential for both appears as soon as an individual tailors. It has been described by all of the aforementioned studies that this potential interdependence and collaboration often results in actual interdependence and collaboration if the technical and organizational framework allows for this.

2.3. Conclusions

All of the above makes collaborative tailoring an interesting field for research: collaborative tailoring may make both the work of a group and its individuals more efficient and create synergetic effects for them. On the other hand, tailoring is an effort that many people cannot or do not want to undertake due to the fact that they think they cannot afford the time, because it is not part of what they are supposed to do or because the software does not provide mechanisms that seem appropriate to them. Collaboration on top of it seems to make it even more difficult to them.

At the heart of tailoring is the question how much and what kind of flexibility a software should provide so that users can benefit from it without being too restricted or overwhelmed with all the options and possibilities available. At the heart of collaboration at the workplace is the question how software and organizational settings should look like to support collaboration adequately.

The previous work described in this section showed different aspects of collaborative tailoring and the fact that it can be a worthwhile activity. However, the presented results were often not generalizable and had not

been brought together. Taking this generalization and combination as a basis for two cases and subsequent suggestions for collaborative tailoring, the thesis intends to show how the positive aspects of collaborative tailoring can outweigh the involved effort and resources and how software and organizational embedment make collaborative tailoring beneficial. The pragmatic approach taken here includes both software design aspects without going into too much detail and aspects of organizational embedment without just staying fuzzy and avoiding detail. The following two cases that I was involved in provided much understanding how to go about such a pragmatic approach.

3. Two Cases

In this section, two cases are presented that each shed a different light on tailoring and collaborative aspects of tailoring. In the word processor case (see third paper of collection), the explicit aim was to support collaborative tailoring of a single user application mainly by the objectification of tailoring files and by providing mechanisms to distribute and share these tailoring files. The groupware search tool case (see fifth paper of collection) has a longer history of development. Here, a strong user involvement over time led from the development of an improved untailorable search tool to a tailorable search tool. The field test of the last version of the groupware search tool that then did include several features to support collaborative tailoring is described in the section *Suggestions for Supporting Collaborative Tailoring*.

3.1. Word Processor

Generic single user applications usually do not provide support to share tailoring files among its users. However, they are often tailored collaboratively. As described above there are several positive experiences with the sharing of tailoring files of single user applications, like word processors or spread sheets. To support such collaborative aspects of tailoring single user applications, an extension to a common off-the-shelf software that should allow the exchange of tailoring files was to be developed. In order to get a deeper understanding of how people collaborate in tailoring a word processor I read the relevant literature described above and a field study was conducted. The result of the study was a number of different collaborative tailoring use situations focusing on the exchange of

document templates and toolbars. Based on an analysis of these use situations a tool, which provided collaborative tailoring functionality was implemented as Microsoft Word 97 extension.

3.1.1. Setting

To learn about users' habits and to inspire the design, a qualitative field study with users of Microsoft Word has been carried out. We conducted 11 semi-structured interviews with users from four different fields (public administration, private company, research institute and home users). Depending on their field of application the interviewees reported about differences in the extent and the way tailoring is seen as a collaborative activity. We identified four typical use situations that showed a variety of collaborative forms to tailor word processors, covering the classification of figure 2. Figure 3 positions them in Oberquell's (1994) matrix (see also third paper).

		Actors	
		Individuals	Group
Persons Affected	Individuals	Experience transfer among insulated home users	Collaborative tailoring and organization-wide distribution
	Group	Central repository for standardized forms	Shared document templates and notification of users

Figure 3: Position of use situations in Oberquell's matrix

Different groups of users were involved in the collaborative tailoring process depending on the respective tasks. Thus, support for collaborative tailoring should allow differentiating among various groups of users when sharing tailoring files. Sharing of tailoring files can require different mechanisms. In cases a power user builds a tailoring file required by a user for the task at hand, a mail tool seems to be the appropriate technical support for distribution. On the other hand, if tailoring files are not required instantly by a specific user, a publicly accessible store allows to select among these tailoring files. The interviews also showed that there is a need to make users aware of the fact that somebody else has produced a tailoring file with relevance to them.

Evaluating the use situations and summing up the results of the final discussion with the interviewees, the following main requirements for the tool emerged. It turned out that this empirical evidence is in line with theoretical and empirical work described in the literature about tailorability:

- tight integration in the word processor application;
- mechanisms for sharing, sending and receiving tailoring files
 - a public store to provide a location to exchange tailoring files;
 - mailing mechanisms for users to be able to send tailoring files directly to other single users and groups of users;
 - a private workspace for tailoring files, that may be copies of files from the public store or files received from others via the mailing mechanism;
- an awareness service which notifies users about modifications of tailoring files.

Consequently, the respective features were provided in a prototype as an extension (“add-in”) to Microsoft Word implemented in Microsoft Visual Basic for Applications. The features included *sharing document templates and toolbars*, *identifying tailoring files in shared workspaces* by means of annotations and a preview mode, and a *notification service* to inform users of the arrival of a new tailoring file in their inbox. Finally, a usability test of this add-in has been conducted by using the method *constructive interaction for testing collaborative systems - CITECS* (see fourth paper of collection).

3.1.2. Evaluation

The evaluation resulted in findings on different levels. Most obvious, there were some shortcomings of the interface resulting in the need to change the names of some buttons. Moreover, all of the participants considered the possibility to save, connect and distribute tailoring files to be very helpful for their work. Although not all participants were expert users they were all able to use the tailoring functionality and the sharing functionality. The overall usability of the tool was perceived to be good. The discussion following the tasks revealed that the participants’ conceptual model of how the distribution of files worked was very close to how we, the designers, had intended and implemented the distribution.

Besides the qualitative usability test a quantitative evaluation in which 32 persons participated was conducted (see third paper). The aim of this quantitative evaluation was to find out how the option to send and receive tailoring files with the extension performs in comparison to the sending mechanism already implemented in Microsoft Word in the file menu. The menu item *Send To* spawns an external email client with an outgoing mail that contains the current Microsoft Word document template as attachment. The hypothesis was that the extension would not rank worse than the internal mailing mechanism even if it was unknown to users. To test this hypothesis 32 persons of at least average computer skills had to test both the extension and the internal mailing mechanism. The means for the functionality and the usability regarding sending and receiving for both the extension and the internal mail mechanism were between 4.5 and 4.9 on a scale from 1 to 6 (very bad to very good) with a maximal difference of 0.2 between the extension and the internal mail mechanism in any given category. Despite the fact that the extension was only an unoptimized prototype with the first version of the user interface the participants could obviously detect the value in the strong integration and the enhanced functionality of the extension.

The word processor case showed that even for single user applications collaborative aspects of tailoring are an issue. It also showed that, with relatively little effort, several important features for collaborative tailoring can be supported: the word processor was already tailorable on several levels (choosing from alternative behavior, creating macros) and extensible by an add-in implemented in Basic; most organizations have their computers connected and already support some forms of shared workspaces. So the technology and infrastructure is mature enough to connect people who tailor single user applications individually in order to be able to introduce collaborative aspects. Moreover, the case showed how fuzzy the borders between use and tailoring are: The tailoring files most interesting to the word processor users were document templates, which are basically write-protected documents. However, as templates they were considered to be tailoring files, since they modified the functionality of the word processor in use, because the users could start the word processor by double-clicking a template and immediately were able to do their task of, say, writing a formal request.

3.2. Groupware Search Tool

The POLITEAM project was a collaborative software development project in which the target organizations required technical support for distributed collaboration. The aim of the POLITEAM project was to develop a system, which supports distributed work in large organizations. This was done by providing a workflow component to handle circulation folders, that structure the workflow, and by implementing the metaphor of a *shared desk* that integrates document processing tools (cf. Klöckner et al. 1995). The application partners of the POLITEAM project were two government organizations: a subsection of a German Federal Ministry and the State Representative Office of a German state, both located in Bonn. In the POLITEAM project *user advocates* played a special role. These project members visited the sites regularly, provided support to the users and thus were able to perceive user requirements in a direct way (cf. Mambrey et al. 1996). For each of the application partners that were to introduce the POLITEAM system into their organization, their work and organizational structure has been analyzed. After configuring the first versions of the POLITEAM system to each of the application partner's needs, it was introduced in their organizations so that about 40 persons altogether worked with the system.

Among the multitude of challenges that the POLITEAM project had to face dealing with such a complex issue as the introduction of groupware in large administrations, searching of files was of particular interest. To meet the respective requirements, in the course of the POLITEAM project a tailorable search tool has been developed for the participating users to search for files in the shared workspace. For the development of the search tool, we proceeded in three major steps. First, Search Tool 1 using Microsoft Visual Basic for Applications was developed to meet general needs with a default setting for a generic search tool (see fifth paper). In a second step, the evaluation of Search Tool 1 was used for a component-based reimplementations in Java. The resulting Search Tool 2 allowed end-users and organizations to tailor the tool to their particular search requirements. Finally, Search Tool 3 provided mechanisms to support collaborative tailoring and the exploration of the tailoring functionality. Search Tool 3 and its usage are described in the section *Suggestions for Supporting Collaborative Tailoring*.

3.2.1. Search Tool 1: Involving Users

The basic version of LinkWorks had a tool implemented that allowed the user to search for any object, independent of its actual location within the system. Discussions with users revealed that this search tool was not well enough designed to be used by the application partners, since the issues of privacy and unintentional manipulation of shared files were not satisfactorily dealt with, possible conflicts about snooping around on others' desks had not been considered.

In the course of the redevelopment of the search tool, different techniques for requirement analysis have been involved. We conducted 10 interviews with interview partners from four different organizations, one of which was a POLiTEAM application partner, and held four workshops with POLiTEAM members, where aspects of searching were raised, two of which were dedicated to search tool prototypes. Moreover, the POLiTEAM user advocates helped us to get a better understanding of the work of the application partners and their requirements.

The search tool prototype was implemented by using the LinkWorks programming interface. A major improvement in the resulting prototype 1 of Search Tool 1 was the distinction of the area where an object was found (i.e. on the searcher's own desk, on someone else's desk or in the archive of the group). This prototype was presented to system developers and user advocates, then changed. The changed version (prototype 2) was shown to three users from one of the application partner organizations in a workshop with the primary aim to evaluate the functionality and user interface of the new search tool.

These users not only suggested some minor changes of the user interface, which were considered in the next prototype, but also hinted at another major feature that could be subject to tailoring: They suggested that objects found by the search tool might not only be represented as a link to the original object on the searcher's desk, but might alternatively be copied from the owner's desk to the searcher's desk. While this might be seen as a contradiction to the design ideas of LinkWorks, it became obvious that this was the appropriate solution for some settings. Though we initially thought that this prototype could become part of the POLiTEAM system, the feedback from the workshop and the statements of the user advocates convinced us to redesign the prototype and provide a tailorable version. Prototype 2 of

Search Tool 1 suited most of the needs of the users of the application partners. In this way, it was considered the default configuration and starting point for a future tailorable system and as a means to have the users get to experience and get used to electronic search.

In a next step, the search tool was enhanced by different mechanisms supporting the choice of functionality options and the construction of conditions for system behavior. To do this, the initial interviews conducted before the first prototype were helpful to identify options for tailorability. The broad approach to get requirements from different organizations by conducting interviews, doing workshops, discussing prototypes including hands-on-experience and being supported by the knowledge of user advocates helped to provide a deeper understanding of the aspects to be considered to develop a tailorable search tool for groupware.

3.2.2. Search Tool 2: Towards Component-Based Tailorability

After the implementation of a search tool that suited the basic needs of a range of users, a component approach to create a tailorable search tool has been applied. By using the established work on components and wiring diagrams to connect these, we wanted to enable users and local experts to create, modify and test search tools during runtime.

Component architectures as a means to design applications have been known for quite a while. In the development of the component-based search tool we focused on the possibilities - not for software programmers and designers but rather for users and local experts - to tailor a search tool during runtime. To implement such a tool, Sun's JAVABEANS (see JavaSoft 1997) component model was used which allows for dynamical binding of components. We provided a layered architecture where several atomic components could be combined to a compound component.

The deconstruction of a search tool into components was one of the foci of a full-day workshop with 9 persons from the Federal Ministry and the State Representative Office. The analysis of several other search tools had shown that a first approach to a composition would be two distinguished parts, related to the chronological sequence of searching for electronic files in a shared workspace. First, a search is performed according to a specific search request, then the results are presented and can be used for further work. The

workshop confirmed this and provided further hints particularly for different in- and output switches taking into consideration e. g. age and name of files.

The division into two parts seen from the user's perspective results in the division into three parts that were actually provided. This is due to the provision of usually invisible components which do the actual work: the so-called flow components are used to perform a search and to have an effect on the search results, like splitting it into two output streams (switch) or retrieving some more of their attributes (e. g. date of creation, last change).

Based on the results of the introductory workshop of Search Tool 2, Search Tool 1 was decomposed into several different components which are divided into three categories: input components, data flow components, and output components. The input components are used to specify the search and to trigger an action. Here, e. g. the name of the objects that are to be found can be entered, and a search can be started.

In order to allow users and local experts to tailor search tools by connecting components, an integrated runtime and tailoring environment has been developed, which serves as basis for the deployment of a component-based application. After the search tool is started, a user can either just use the search tool or enter the tailoring mode where the components and their wiring are shown. The components can be deleted or modified, new components can be added and the wiring can be changed as long as it obeys the wiring rules.

To evaluate the design of the component-based search tool and its integrated runtime and tailoring environment, we held another workshop where several aspects of Search Tool 2 were discussed.

Understanding component architectures

Search Tool 2 consists of several components as well as a runtime and tailoring environment. This approach was clear to all users. The layered architecture was understood only by the more experienced users.

Additional components

Several requirements for additional components emerged during the workshop. Some users requested the possibility for a full text search, others asked for more expressive display components and the possibility to access

the found objects directly rather than having to create a copy or link, which they would have to open via another window. The users also asked for more switches, e.g. a switch which displayed all the documents older than three months in a window, separated from a window showing documents, which had been written more recently, and a switch which allows to display objects in different windows, depending on the location within one person's folders.

User interface

After the presentation, the users stated that the graphical interface for connecting components was too complicated to handle without additional support. The following discussion in the workshop resulted in three suggestions. Firstly, the users asked for a better description and for more appropriate names for the different components they could select from. Secondly, they argued for a context sensitive quick-info delivering more comprehensive explanations about the behavior of individual components or search tool alternatives. Thirdly, they asked for a context sensitive behavior of the select box in which all components available in the system were listed.

Collaborative aspects

During the workshop, there was also a discussion about collaborative aspects of tailoring search tools, particularly about the division of labor within the group. Some users argued that they usually had not the time to tailor a search tool and suggested that the person, who already provided local support, should create different search tool versions, so the others could just select a search tool from different alternatives. The user providing local support was enthusiastic about the tailoring environment and suggested that we should provide a tool to distribute the tailored search tools among the group. He argued that his job would become much easier with such a tool.

Exploration

The complexity of the search tools and their wiring makes it difficult to know if a new search tool really does what a user wants it to do. Therefore, the user working on local system support asked for some way to test a search tool. In order to fully perceive what the search tool does and not to

disturb other users and their privacy by searching just to test a search tool, a test environment with artificially created data could encourage tailoring. Such a test environment would also prevent the users from unintended deletion or manipulation of “real” data.

3.3. Review of the Cases

The two presented cases show different approaches to, and highlight different aspects of tailorability. In the word processor case, the starting point was the idea to support collaborative tailoring of a single user application by allowing objectification, sharing and sending of tailoring files. This idea has been realized in a straightforward way: The initial understanding of collaborative tailoring was enhanced by interviews in several settings about collaborative tailoring which resulted in the description of typical use situations. This in turn led to the implementation of an add-in to Microsoft Word, which was tested in a laboratory setting. In the case of the groupware search tool, the tailorable search tool evolved as part of a long process of introducing groupware in “real life”, i.e. in two subunits of the public administration. There was much interaction with users involved e.g. through interviews, workshops and user advocates. In addition, the process included more iterations and successive prototypes. So with this much broader approach the process was slower but also deeper: the resulting solution was already suited for a particular group, which had begun to accept the search tool certainly because of its features, but also because it had been brought to them in a process that they had participated in.

Concerning collaborative tailoring, both cases complement each other and the literature: The interviews in the word processor case clearly show, that even for single user software like work processors, collaborative tailoring does exist. Some of it is not even supported by a computer network, e.g. looking how something is tailored and trying to repeat it at home, or passing floppy disks. The implementation shows the general possibility to enhance already existing tailoring possibilities by features for collaboration using the existing local area network. The laboratory evaluation provides evidence that the concept of sharing and sending tailoring files in analogy to other files can be understood and that it may be useful for people’s work. The groupware search tool case, as it has been described, above does not go so far in the implementation of different functions. The dynamism of a real

organization, however, is better captured so that the practical use including organizational structures, like the division of labor between the local expert and the other users, becomes clearer.

Together with the literature, these two cases provide the material for suggestions to support collaborative tailoring.

4. Suggestions for Supporting Collaborative Tailoring

This section describes the continued work on the word processor case and groupware search tool case and presents the related suggestions for collaborative tailoring. The work on the cases as described in this section and the development of the eight suggestions for collaborative tailoring have been performed in parallel: on one hand, the continued work described in the cases, including the technical implementations and the field test, reached a level of maturity that made the suggestions possible. On the other hand, a provisional conception of what would be beneficial for collaborative tailoring was derived from previous work on the cases and the literature and guided the implementations and the field test. Thus, the continued cases and the eight suggestions were mutually influential. The field test for the groupware search tool and its included tailoring environment also provided some information on selected aspects of collaborative tailoring. For a more detailed description of parts of the first phase but particularly of the second phase of the groupware search tool case, see the sixth paper of the collection in the second part of this thesis. For a more detailed description of the eight suggestions, see the seventh paper of the collection in the second part of this thesis.

The suggestions are the quintessence of my work on collaborative tailoring in the last years and my contribution to the ongoing discussion. They are the result of an intensive literature study and collection of descriptions of related work, of my experience as action researcher particularly with tailoring and collaborative tailoring, of the identification and implementation of identified relevant features in software and their evaluation in laboratory settings, and of the field test. The eight suggestions identify the most relevant means by which collaborative tailoring can be supported. They are (1) *provide objectification*, (2) *allow sharing of tailoring files*, (3) *allow browsing through tailoring files*, (4) *provide awareness of tailoring activities*, (5) *make annotations and automatic*

descriptions possible, (6) allow for exploration of a tailoring file, (7) make administration and coordination easy, and (8) support a tailoring culture. They aim at different aspects of collaboration, sometimes showing all or some of the background they originally derive from (e.g. individual or joint file organization for suggestions 3 and 6, CSCW for suggestion 4, or socio-technical considerations from the information systems field for suggestion 8).

4.1. Continuing the Cases

The work on both the word processor case and the groupware search tool case was continued in a second phase following the first phase described in section *Two Cases*. The second phase for each case includes an advanced implementation and a field test.

Note, that the field test for the groupware search tool is completed, whereas the features of the word processor extension are implemented and the layout of the field test is planned, but the actual field test has not yet taken place.

While a distinct list of suggestions to support collaborative tailoring did not exist when the second phase of implementations was started, it is still possible to attribute the resulting suggestions to features of the implementations of the word processor extension and the groupware search tool tailoring environment in the second phase. The following table 1 gives an overview of how this attribution can be made in retrospect.

suggestion	word processor	groupware search tool
1 Objectification	document templates	components and whole tools
2 Sharing	shared workspace and sending / receiving	joint pools for tools and modules
3 Browsing	in private and public workspace	in pools for tools and modules
4 Awareness	notification of tailoring file in inbox	not for collaborative parts

5 Annotations & Descriptions	both supported	both supported
6 Exploration	not in this version	exploration on simulated data supported
7 Administration & Coordination	administrator has special rights; keywords for tailoring files	everyone can manipulate tailoring files
8 Tailoring Culture	supported by workshop and input of relevant tailoring files	supported by workshop and visits to encourage tailoring

Table 1: Attribution of suggestions for collaborative tailoring to second phases of the two cases

4.1.1. Word Processor

Based on the experiences from the previous evaluation of the prototype of the word processor extension in the first phase (see section *Two Cases*), a new version of the extension of a common word processor supporting users to tailor collaboratively in several ways has been built. The extension was completely re-implemented to ensure the necessary stability. In continuation of the word processor case this new extension will be tested under real working conditions for several weeks by a group of 5-15 people that still has to be identified. At the end of the usage period, the participants will be interviewed about their use of the extension. The use of the extension and the related collaborative tailoring activities will then be logged and the resulting tailoring files will be gathered and analyzed (permission of participants provided).

The extension and the respective field test incorporate aspects of almost all of the suggestions for collaborative tailoring as indicated in table 1.

4.1.2. *Groupware Search Tool*

Also, a new version of a tailorable component-based groupware search tool has been created which incorporated the support of several aspects of collaborative tailoring in a tailoring environment.

This new Search Tool 3 (see section *Two Cases* for a description of the preceding versions Search Tool 1 and 2) was used in a field test with three users in the State Representative Office of a German State that had already been part of the action research involved with Search Tool 1 and 2. Other users of the State Representative were asked to provide search permission on their documents eventually needed by the participants of the field test (see sixth paper for details on the development and evaluation of the component-based tailorable search tool and the tailoring environment).

Note the methodological weakness that the field test of Search Tool 3 in the State Representative Office contributes to a basic form of evaluation of the suggestions for collaborative tailoring, although the use in the same organization had already contributed to the development of these suggestions. This weakness is justified by the fact, that the development of such a complex tool, its relation to the State Representative's work practice based on action research, and the subsequent acceptance of the researchers involved as well as the willingness to use the tool within the organization, was the result of long and intensive work. There were no resources available to create this understanding of another organization and the acceptance there to use this rather complex groupware search tool to be able to do the field test in another organization.

We presented the new search tool environment in a workshop in which three users, one user advocate and three designers participated. In the following two weeks, the search tool environment was applied in a field test and the users were supported continuously by a researcher. A researcher also visited each user at least twice for a 60 - 120 minutes time span for personal support and in order to encourage them to tailor. The tailoring process and the emerging problems have been observed and the respective written notes were transcribed.

At the end of this observation period, a last extension of the search tool was introduced that included a simulated search in the groupware environment. Several days after installing this new version, semi-structured interviews

were carried out with the users. The interviews covered issues related to the tailoring environment: patterns of collaborative tailoring, usage of textual documentation (manuals, help functions, annotations), occasions and means to experiment with applications, and further design requirements. The interviews lasted about 60 minutes and were carried out at the users' workplaces.

Search Tool 3 incorporated aspects of almost all of the suggestions for collaborative tailoring as indicated in table 1.

4.2. Eight Suggestions

The second phases of the cases contributed much to the suggestions for collaborative tailoring. However, the conduct of the cases themselves and the eight suggestions were also heavily influenced by existing work on tailoring and collaborative tailoring.

One of the preconditions for collaborative tailoring is that a system can be tailored at all. There is a large body of literature on the question of how tailoring of a single user application can be supported (see e. g. Oppermann 1994). Some of the following suggestions are inspired by this. However, tailorability of the application underlying the collaborative tailoring efforts is assumed. Another source of inspiration is the discussion of Computer Supported Cooperative Work (CSCW) in general. The suggestions presented below focus this discussion to the particular topic of tailoring. Undoubtedly, CSCW as a research field will be a constant source for improved and new suggestions for supporting collaborative tailoring. The suggestions for supporting collaborative tailoring are presented at length in the seventh paper. They may serve for researchers to refine and being transferred to other areas, for software developers to have a guideline for implementing necessary functionality and provide adequate software structures, and for practitioners to be able to select and tailor generic software and provide organizational structures that support collaborative tailoring.

The following suggestions concern both technical and socio-organizational aspects of collaborative tailoring. It is important to note that neither technical nor socio-organizational measures to support collaborative tailoring alone suffice: they must be combined. Often the distinction is fuzzy and a particular suggestion belongs to both areas to a certain extent. In

the following, first the suggestions that are located more in the technical area, after this, socio-organizational measures are proposed.

4.2.1. Suggestion 1: Provide Objectification

A prerequisite for most forms of collaborative tailoring is that tailoring is made persistent in tailoring files. This *objectification* (Henderson & Kyng 1991, p. 232) allows users or administrators to access, modify or share the tailoring files by the usual means that files are processed. Thus, the already existing infrastructure for file processing like the operating system, the network and applications can be used. Objectification not only allows users an easier technical approach to the tailoring files. It may also have the psychological advantage of moving tailoring closer to using since tailoring then is no longer associated with an application that is difficult to change but may be considered on a higher level but similar to the files associated to and modified by the application. Looking at figure 2 in section *Foundations & Related Work*, the objectification particularly supports those forms of collaborative tailoring where the actor and the affected person are different. In this case, the objectified tailoring activity can be easily exchanged. Note, that objectification is usually, but not necessarily, the prerequisite for different forms and modes of tailoring:

- a tailoring activity to a word processor that does not allow an objectification of the tailoring activity in a file can be repeated on a second computer on request of an interested colleague;
- a groupware that allows tailoring but not an objectification of the tailoring activity in a file still permits collaborative tailoring e.g. where several persons tailor the groupware with effects for the whole group;
- tailoring activities may also occur or be exchanged by means of a network in other forms than files, e.g. in a tailoring stream sent and received through ports at specified locations and occasions.

In both phases of the word processor case, document templates as most important tailoring files, which also include macro information, are objectified in a file. The reasonable objectification of other possible tailoring files of interest, like toolbars or directly manipulable macros, is not provided by the word processor in the second phase and would require much work dealing with the word processor's internal features and data formats.

In the groupware search tool case, the layered component-based approach provided an ideal basis for the objectification of tailoring files. In the first phase, it enabled us to have components of different granularity (atomic, composed, complete search tools) as objectified tailoring files. In the second phase, components and whole search tools could be stored separately. Complete tailored search tools and modules combined of basic components could be kept in a common directory of the groupware. Their particular file structure allowed to add the saved search tools and modules directly to a combo box (the *tailoring box*) containing a list of search tools or modules from which new tailoring files could be created. It was accessible for all users from the tailoring menu.

4.2.2. Suggestion 2: Allow Sharing of Tailoring Files

Sharing tailoring files or ideas with others is one of the most common and powerful forms of collaborative tailoring. The interviews in the word processor case (see third paper) showed how important and common sharing of tailoring files is as a form of collaborative tailoring. Sharing tailoring files is often supported in two ways: Either a shared workspace is provided to keep tailoring files for a group to retrieve, or mechanisms for sending or receiving tailoring files are provided. In analogy to the treatment of other files that are handled collaboratively and depending on the concrete task and organizational setting both, the centralized and the decentralized form have advantages. If there are already shared workspaces for other files existing, a particular shared workspace for tailoring files can be very useful, because the group is used to working with shared workspaces and there is less trouble with different versions of a file. However, if a certain number of files has been reached, the need for some administration of the shared workspace arises. The decentralized approach of sending and receiving tailoring files leaves the administration to the individuals and bears the danger of having numerous and possibly inconsistent local copies of tailoring files. Again in analogy to the treatment of other files, it is recommendable to provide support for both forms of sharing tailoring files and support the users to let effective and efficient ways of usage emerge.

In the case of the word processor, several mechanisms for sharing, sending and receiving tailoring files were provided (see section *Two Cases*): a shared workspace to provide a location to store tailoring files, mailing mechanisms for users to be able to send tailoring files directly to other

single users and groups of users, and a private workspace for tailoring files that may be copies of files from the public store or files received from others via the mailing mechanism. The laboratory test of the first phase of the word processor case showed that this distinction had been understood (see fourth paper). In the second phase of the word processor case, a folder substructure is added to allow for more detailed filing with the option to use a keyword system for tailoring files.

The groupware search tool tailoring environment allowed users to share tailoring files by saving them in a shared workspace. There, they could be deleted, renamed or copied (see sixth paper). Due to the file structure and its presentation in the tailoring box (see suggestion 1) new tailoring files were immediately shared with all other users. The fact that a form of sharing was provided was welcomed by the users. They did not consider it a problem that a tailoring file they produced was immediately available to others. One user argued that he was interested in an additional personal space for tailoring files so others could not observe him tailoring since originally he thought this might not be considered important work that he was supposed to do.

In the literature, too, several authors report on different forms of sharing tailoring files (see also section *Foundations & Related Work*). Mackay (1990) reports on various forms and preconditions of sharing tailoring files, ranging from telling someone how to tailor to reach a certain effect to actively putting a tailoring file into a predefined shared workspace. MacLean et al. (1990) explicitly support sending tailoring files to others with their Buttons system. Wasserschaff & Bentley provide mechanisms that their Tviews (tailoring files describing a particular view on a document) can be distributed via the groupware system that they have been programmed for. In the discussion about collaborative tailoring sharing tailoring files is considered a necessity to support collaboration. Henderson & Kyng (1991, p. 233) state that “*means must be available to acquire changes*”; Bentley & Dourish (1995, p. 145) require that it be “*possible to add attachments to the shared workspace for others to retrieve and use*”.

4.2.3. *Suggestion 3: Allow Browsing Through Tailoring Files*

Browsing helps people to find information they need even if they may not look for a specific information. Particularly for tailoring files, it provides

support on two levels. Firstly, users browsing through tailoring files may find files that they are interested in functions that they had not thought of before. Secondly, on a higher level browsing through tailoring files is a means to get an overview over the tailoring of a group particularly for new group members. By browsing, they can get an impression of the number of files, the structure of folders possibly representing different areas of tailoring, and the annotations and automatic descriptions which provide information about active tailors and their tailoring focus.

The interviews in the first phase of the word processor case and the experience in the groupware search tool case revealed that users are willing and interested to use others' tailoring files but often do not know, if there exists a tailoring file that may meet their needs. This corresponds with the observations of Mackay (1990) on patterns of sharing tailoring files where several of the methods to obtain or give tailoring files support the notion of vague initial needs satisfied by asking others and stepwise finding out what it really is that you want to know. Consequently, Mackay concludes that tailorable software should provide the ability to browse through others' useful ideas.

In the first phase of the word processor case, the possibility to browse through tailoring files including the display of annotations had been provided (see suggestion 5: *Make Annotations and Automatic Descriptions Possible*). In the second phase, the main window of the word processor extension additionally allows browsing through both the file hierarchy of the shared workspace and for every single person browsing through his or her personal workspace.

In the second phase of the groupware search tool case, the lists of the search tools or modules in the tailoring box could also be browsed. Here, the names of the files as given by the creator gave first hints to the functionality and appearance of the search tool or module. During the field test, tailoring files had been created and deleted but the list of search tools never contained much more than ten items which made browsing easy, and no additional structure e. g. by subfolders was necessary.

Technically browsing can be supported by adequate browsing facilities that can give both a general impression about what files are available and, in connection with other features like annotations (see suggestion 5), can provide relevant information, e. g. a brief description of the functionality at a

quick glance or the name, creator, date of creation, size, number of people who downloaded or saved it from the shared workspace, or an average of all marks given to that tailoring file by all users who tried it out. Additionally, different sorting mechanisms for tailoring files in a directory can give an overview. Bearing in mind that collaborative tailoring currently is rather an activity of small groups, lightweight solutions seem to be preferable to very elaborate browsing tools that may incorporate advanced database technology but are difficult to handle and require much input by the creator of the tailoring file.

Interestingly enough, the concept of general browsing has had an enormous upswing with the growth of the World-Wide Web. On one hand, this is obvious considering the hyperlink structure of the WWW. However, the sheer mass of files in the WWW seems to make finding strategies more adequate that contain strong aspects of direct logical or keyword search with a search engine, whereas general browsing seems a more suitable concept for small or medium sized collections of files that may be pre-structured by a folder structure or keywords or any form of subject trees.

4.2.4. Suggestion 4: Provide Awareness of Tailoring Activities

Collaboration needs information about what others do. This also concerns the need for awareness about others' tailoring. One way to create this awareness is by notifying others of the existence of tailoring activities or tailoring files. Henderson & Kyng (1991, p. 233) claim that "*news must be published that change is available*". This helps users to stay up-to-date and avoids double work on the same tailoring issue. If tailoring files are shared this can be accomplished by a simple event driven notification service stating that a person received a tailoring file from someone else or that someone has uploaded a new tailoring file in the shared workspace.

Several interviewees in the word processor case reported the importance and different forms of making others aware of their tailoring work, like just telling them or putting a notice on a non-electronic blackboard. In the word processor case, awareness about others' tailoring is provided by a notification service that is triggered whenever a tailoring file arrives in the tailoring inbox. The notification service informs the user via a message window at start-up time of the word processor and when the user activates a tailoring function in the menu. This window presents the tailoring file and asks the user either to store it in his or her private repository or to delete it

instantly. Another simple form of awareness is provided by the fact that the browser for tailoring files always shows the list of tailoring files in the shared workspace so that new files can be recognized.

This direct form of informing others that someone is engaged in tailoring is only one of many ways that awareness can support collaboration. The term and notion of *awareness* have been very popular in CSCW research in the last couple of years since awareness is considered to capture many aspects of what makes collaboration successful. The basic definition is provided by Dourish & Bellotti (1992, p. 107) who define awareness as being “*an understanding of the activities of others which provides a context for your own activity*”. Gutwin (1997) distinguishes *Informal Awareness*, *Conversational Awareness*, *Structural Awareness*, and *Workspace Awareness*. Among other things, they serve for recognizing opportunities for collaboration, relate behavior in a conversation, relate behavior to the knowledge of a group’s organization and of the working relationships within it, and concern people’s interaction with the workspace. All aspects of awareness in the general context of collaboration can be applied for collaboration in the context of tailoring. In the latter case, awareness gains a particular importance since tailoring for most people is not their primary work task. Therefore, they might not actively be concerned about informing themselves about others’ tailoring and need particular mechanisms to be kept informed. Note, that annotations (see below) can be considered to provide awareness by making context of the tailoring explicit.

4.2.5. Suggestion 5: Make Annotations and Automatic Descriptions Possible

For collaboration, it is helpful to understand the context in which the other participants work. As described above, creating awareness is one means to provide understanding of context. In the groupware search tool case, it became clear after a while that it was difficult for others to understand what a particular search tool created by one person did. Therefore, the participants required more context by a textual description. Such annotations serve to enhance learning and share that understanding with others (Henry 1997). Active annotations, i.e. adding a critical or explanatory note to a file, are a good means of providing context particularly to a shared file. This is more necessary for a tailoring file than for a general file concerning the primary work task, since the tailoring file is

usually less self-explanatory and has a form that is less known to users than the files they usually work with. Here, annotations can serve to explain the context and the function of the tailoring file. Similar to remarks added by programmers to the program code they write, annotations added to tailoring files by tailors serve to provide a better understanding of what the tailoring file's function is. Annotations can be made as plain text provided by the tailor to go with the tailoring file. Similar forms of creating context are the selection from a list of categories or keywords that can be associated with the tailoring file or even automatically generated descriptions of parts of the context that the tailoring file was produced in, e. g. name of the tailor or date and time of tailoring. Mørch (1995b) provides context in a similar way by a presentation of the *rationale* in his layered architecture for tailorable applications.

In the main window of the word processor extension, the annotations provided by the creator of the tailoring file and automatic descriptions like author, data of creation and name of included macro collection could be seen in both phases of the word processor case.

The annotations and automatic descriptions for the tools and modules could be directly accessed from the tailoring box in the second phase of the groupware search tool case. A plain text annotation could be provided by the creator of the search tool or module, who could also decide to provide her or his name in the respective field. The automatic description contained versioning information in case the search tool or module was based on an existing search tool or module. If the name of the tailoring file has been changed, the existing annotations and automatic descriptions were kept as a changeable default. As an additional feature the help texts for the tailoring environment were available in a shared workspace and could also be annotated so that people could read others' annotations to the help text.

All users in the groupware search tool field test considered annotations and automatic descriptions to be very helpful. They argued that these helped a lot to understand the search tools and modules and could make the exploration easier or even completely obsolete. All users were willing to write annotations but generally unsure about what to write. Accordingly, users sometimes had difficulties to understand what other users' generally brief annotations meant. Additionally, the usage of the annotations was different: one user described what the tailoring file did, another described the internal wiring and someone else suggested it would be helpful to

describe what a module or search tool was made for. The option to name the creator of a tailoring file was positively received: this could help to talk to the person in case of difficulties but also to judge on the quality of a tailoring file. Since the version history was never longer than one file in the field test, it did not play an important role. The help texts were rarely used in the field test and never annotated.

4.2.6. *Suggestion 6: Allow for Exploration of a Tailoring File*

Tailoring files may include aspects of presentation, e.g. a corporate letterhead, of manipulation, e.g. a toolbar for special tasks, or of action, e.g. in a macro. Even with context provided e.g. by annotations, it is not always clear to other users what effects a tailoring file has, particularly if it includes actions or a set of different tailoring aspects. One way to support understanding here is to provide means for *exploration*, i.e. finding out what a tailoring file “does” without necessarily producing all the effects of the tailoring file persistently, and thus avoiding the danger of deleting previous work by some unforeseen effect of somebody else’s tailoring file. According to Engelskirchen (2000) this includes several mechanisms, like undoing the effects of a tailoring file, working in a neutral mode, where the effect is shown but not persistent, or trying out the tailoring file on test data that can be changed or deleted without harm.

In the search tool case, exploration was a critical issue, because the user acting as local expert asked to be able to test newly created search tools since he wanted to know whether the tailored search tools did what they were supposed to do. In the second phase of the groupware search tool case, exploration of a search tool was possible on simulated data. The exploration mode was visualized by a red background and allowed trying out a search tool on simulated data rather than real data to avoid unintended data manipulation and provide full control to the exploring person over the data space to search.

The users liked the exploration mode because it provides complete security from irreparable damage to the system they had experienced with other software and particularly its insufficient undo function. They considered the visual closeness to the real mode to be very helpful for their understanding and the red background sufficient to be aware of acting in the exploration mode. The usage of simulated data was basically understood by the users, but considered to be rather time-consuming due to the need to create

simulated data for particular test situations. They recognized the search on simulated data as one way of understanding the behavior of a search tool on another person's desk. The effect of searching on simulated data to understand how a search tool works is similar to the analysis of the search tools' or modules' components or the discussion with colleagues, but avoids some of the disadvantages of these approaches. All in all the users considered the search on simulated data as adequate for users on an intermediate expertise level who know enough of the search tools to understand the basics but still need assistance.

4.2.7. Suggestion 7: Make Administration and Coordination Easy

Technical and organizational means should be provided to administrate and coordinate the tailoring activities of a group. Organizational means are described in suggestion 8: *Support a Tailoring Culture*. For the technical means, there is again a strong analogy with the administration of general files and the coordination of general activities in collaborative settings.

The administration of tailoring files is necessary particularly with a growing number of tailors and tailoring files. The administrator is a person who has an overview over tailoring and is able to order the tailoring files. The administration may include taking care of a shared workspace by removing old versions or by checking and debugging files that are put into the shared workspace. The administrator may also compare and combine several tailoring files. This requires administrative access rights and possibly the installation of a tailoring *sandbox* where a tailoring file can be tested (see also suggestion 6).

The interviews in the first phase of the word processor case revealed the importance of such administration and coordination for several interviewees e.g. in relation to organization-wide document templates for administrative purposes. In the second phase of the word processor case, the role of an administrator who is allowed to delete files from the shared workspace is provided for the shared workspace. Tailoring files also can be attributed by given key words to provide additional structure. For the planned field test, coordination of the tailoring activities will be provided by a researcher.

In the second phase of the search tool case, such an administrative measure was provided, too: The tailoring files could be manipulated (i.e. copied, renamed, deleted) in the tailoring box by everyone. During the field test

coordination of the tailoring activities (suggestion 7) was provided by a researcher. The lack of a refined access policy was not considered a problem since the users felt that the tailoring files and issues should be public – except for the amount of time a person spent tailoring.

Administration and coordination play a role also in the literature on tailoring and collaboration (see also section *Foundations & Related Work*). Henderson & Kyng (1991) suggest that systems used together demand that tailoring should be coordinated. Computer supported coordination of collaborative work in general has been intensively researched and appropriate mechanisms have been proposed (Carstensen 1996). The coordination of tailoring may involve a cross-section of technical mechanisms and organizational measures to support efficient collaboration. While an administrator focuses on work with tailoring files, the coordinator (which may be the same person) focuses on the collaboration of different users and groups. Tailoring files can be considered to be artifacts around which collaboration evolves and subsequently coordination is necessary. The coordination may include eliciting and realizing technical and organizational requirements to support collaborative tailoring. Additionally, an active information policy about tailoring e.g. by email, or the organization of workshops or a mentor system concerning collaborative tailoring can play an important role.

4.2.8. Suggestion 8: Support a Tailoring Culture

Besides and in addition to technical measures the success of collaborative tailoring also depends on socio-organizational aspects and what Carter & Henderson (1990) call *tailoring culture*. The necessity to understand that tailoring and particularly collaborative tailoring can bring benefits to a group working together or to an organization was a prerequisite for our application partner in the search tool case (see fifth and sixth paper) to identify one colleague to be responsible for coordinating and administering collaborative tailoring. The interviews in the first phase of the word processor case revealed several forms of an emerged or installed tailoring culture ranging from a person-to-person exchange of tailoring files or ideas to persons acting as local experts, some of them being officially recognized. The interviews also show that tailoring culture is often developed in a bottom-up process where few persons tailor, then exchange ideas and files, a knowledgeable and interested person emerges as local expert, and ideally

these efforts are institutionalized by officially recognizing the tailoring activities or nominating someone officially responsible for coordinating and administering collaborative tailoring. These, often subsequent, steps represent three possibly coexisting levels of tailoring culture:

- level of equals, people helping each other (see also Gantt & Nardi 1992), or a network of ‘who asks whom’ (Trigg & Bødker 1994);
- different levels of expertise, existence of local experts, “gardeners and gurus” (Gantt & Nardi 1992)
- organizational embedment with tailoring as a community effort (see MacLean et al. 1990) and official recognition of tailoring activities and their importance.

In the second phase of the word processor case, the development of a tailoring culture is supported in several ways: before the introduction of the extension, a workshop will be held where the extension and its benefits are introduced to the participants who also receive a reference manual. During the field test, several tailoring files that have been proved to be of interest for the participants will be created by a researcher, then stored in the public workspace or sent to a participant with the suggestion to distribute them.

In order to support the tailoring culture during the field test in the second phase of the groupware search tool case, after an initial workshop on the groupware search tool, its tailoring environment, and collaborative tailoring, a researcher regularly visited the site and encouraged users to tailor. One user who previously had already shown particular interest, began to grow into the role of a local expert.

4.3. Discussion of Field Test

The results of the field test of the groupware search tool with its included tailoring environment show that the suggestions are in fact useful for realizing support of collaborative tailoring. With the exception of awareness, all suggestions were explicitly realized in the tailoring environment in one form or another and earned their merits in the field test. Awareness was implicitly realized by the fact that a new search tool caused a new menu entry. This low level of obtrusiveness was appropriate considering the relative high effort that the production of a new search tool required and the resulting moderate number of tailoring files produced. The

suggestions concerning objectification, sharing, browsing, and administration and coordination were tightly integrated with the features of the tailoring environment already available for non-collaborative tailoring: tailoring files were just added to a list and access was equal for everyone. This tight integration eased the users' work but may cause irritation in larger groups with frequent changes in the list of tailoring files. For such groups more structure for the tailoring files and additional awareness may be needed, as realized in the word processor extension to be evaluated, where private and public areas are distinguished and different access rights, including an administrator role, exist.

Due to the complex nature of searching in a groupware setting, much effort has been put into realizing features to help users understand others' tailoring files in the groupware search tool tailoring environment. Hence, annotations and exploration were in the focus there. Annotations are generally a good means to create additional understanding. However, they also show shortcomings. Firstly, it is time-consuming to write them and the benefit is not on the writer's but on the reader's side. Secondly, free form annotations can relate to many aspects ranging from the tailoring rationale to technical or user interface issues. While in many cases it is important to allow these different aspects, sometimes it may be helpful to give more structure to annotations, e. g. by providing several input fields for plain text for different issues or by supporting organizational conventions on how to write annotations. In addition, annotations for help texts suffer from the fact that even help texts are not read very often. For the word processor extension, annotations may serve to hint at hidden features of the tailoring file, e. g. macros contained in a document template. Supporting exploration of tailored groupware search tools, particularly on simulated data, is a complex affair. This complexity is due to the nature of groupware, where the effects of a person's action to the work and data of other persons are not always completely visible or understandable to him or her due to access restrictions and asynchronous work. The field test for the groupware search tool tailoring environment shows that this complexity is accepted in order to understand how a particular search tool works by users on an intermediate level. In cases of single-user software, like a word processor, exploration is much less complex and provides a relatively quick and easy way to see the effects of a tailoring file.

In the groupware search tool field test there was not enough time for the group to develop a stable tailoring culture. However, a group member began

to emerge as a local expert. This experience supports the presumption that in a situation of continued use of tailorable software and with forms of aid and stimulation provided for a limited time, as described above, a tailoring culture is likely to develop if the organization recognizes the importance of collaborative tailoring.

The two realizations of the suggestions for collaborative tailoring show two of many ways to put the suggestions into practice. Which of these ways should be taken and which weight and significance is contributed to which of the suggestions depends on the particular setting. Important factors are the software that is tailored (e.g. single-user software or groupware), the concerned organization and organizational embedment of the software (e.g. large or small group involved and issues of organizational distribution of those involved) and the potential benefit from collaborative tailoring.

Certainly, a longer period of usage for both realizations of the suggestions for collaborative tailoring would allow the refinement of the implementation and provide important information on the dynamism of software features supporting collaborative tailoring and their usage as well as the emergence and development of a tailoring culture and their relation to each other. However, the field test of the groupware search tool tailoring environment already underlined the relevance of the introduced suggestions for collaborative tailoring.

5. Conclusion

While it is well known for quite some time, that tailoring activities are often carried out collaboratively, there is a lack of support for this. The aim of my work described here was to provide an answer to the question how collaborative tailoring can be supported adequately. To reach this goal, I performed a survey of relevant literature (see section *Foundations & Related Work*), researched and presented two cases of forms of collaborative tailoring involving action research and the implementation and different forms of evaluation of prototypes (see section *Two Cases*). This lead to suggestions to support different aspects of collaborative tailoring (see section *Suggestions for Supporting Collaborative Tailoring*). The suggestions ranged from software features to socio-technical issues and related both. In parallel, aspects of these suggestions were realized in

software and used in a field test (also see section *Suggestions for Supporting Collaborative Tailoring*).

5.1. Results

The work leading to the proposed suggestions provided a rich picture of many aspects of collaborative tailoring. This picture was then used for constructive suggestions for supporting collaborative tailoring. Necessarily the suggestions are an abstraction of the numerous concrete experiences from the preceding work. The field test of the suggestions shows the validity of this abstraction.

The suggestions cover a broad range of software features and socio-organizational aspects. As can be seen from the two examples and different details described in the section *Suggestions for Supporting Collaborative Tailoring*, the concrete realization of the suggestions can and must differ in different settings. The proposed suggestions may be realized in again other ways where other organizational circumstances or tasks or systems prevail. Advancements in technology may lead to the extension of the list of suggestions or to the replacement of a suggestion by a new one, that may better reflect the changed setting. For each realization of the suggestions this requires a careful reflection of the concrete circumstances of the software used, the tasks performed with it and the group and organizational structure.

One main result of the work at hand is, that tailoring needs a strong integration with the primary work task both technically and organizationally. Only if tailoring and collaborative tailoring are not considered to be completely separate from use they can be successful. Here, further research on adequate software architecture and the organizational embedment of collaborative tailoring is needed.

The presented results suggest that collaborative tailoring may also serve as a medium to encourage groups or organizations to discuss group standards and conventions and thus contribute not only to a tailoring culture but to a general group or organizational culture.

Again, one must bear in mind that this eventually rewarding activity of collaborative tailoring requires willingness and patience. As always, the question remains open how much administrative work the participating individuals are willing to contribute for the benefit of a group or organization and how much administrative effort is still reasonable to stay

on the profitable side of collaborative tailoring. More refined tools to measure this and more refined categories to weigh the individual and group pains and gains against each other are needed.

Now, do the suggestions for supporting collaborative tailoring really answer the question how collaborative tailoring can be supported adequately? The answer is “yes and no”. Yes, the suggestions are the result of careful and comprehensive research where different methods have been employed to understand people’s collaborative tailoring habits and their work practice and to provide, test and evaluate software to support aspects of collaborative tailoring. Therefore, the proposed suggestions are the quintessence of serious efforts to understand collaborative tailoring practice and derive constructive statements from this on a level that is abstract enough, to reflect the analytical content and concrete enough, to provide valuable help for real settings. But no, the proposed suggestions do not claim to guarantee successful collaborative tailoring nor can they cover all of the different circumstances and settings for collaborative tailoring now and in the future. All things considered, the suggestions for collaborative tailoring in hand provide one spin on the upward spiral towards sufficient theoretical and practical knowledge about adequate support for collaborative tailoring.

5.2. Future Work

One starting point for this thesis was the notion that mainly descriptive work about collaborative tailoring had existed and that there had been a lack of constructive results from these descriptions. Now that the suggestions for supporting collaborative tailoring provide such a constructive result and different realizations of the suggestions may be put into practice, it is time to observe again and find out if and how the suggestions can be refined, and if there are hints for a reasonable taxonomy of which aspect of collaborative tailoring has which impact in a particular setting. Longer term observations could also provide information about technical and organizational dynamism going along with collaborative tailoring and how this in turn may influence which aspects of collaborative tailoring are most relevant.

To support collaborative tailoring and the integration of work task and tailoring a generic tailoring organizer belonging to different applications could be helpful. This organizer could combine mail mechanisms with the operating systems’ functionality for access rights or shared workspaces and an enhanced functionality for awareness and annotation. First steps towards

this direction are taken by work dealing with component architectures for tailorability of groupware.

Right now, it seems that tailoring is most useful in small groups with similar work tasks and personal acquaintance. Future work should also address the question of technical and organizational scalability of collaborative tailoring. For larger groups of participants the distinctions between public and private spaces and between creator and user of tailoring files need to be enhanced. Like in shared workspaces for general purpose, a more sophisticated access control model is needed. Another form of supporting collaborative tailoring would be to allow the worldwide distribution of tailoring files, e.g., via the World Wide Web (WWW). Thus, one could even think of supporting global teams or even establish widely accessible libraries for tailoring files. Whether this is, however, reasonable in the light of the poverty of organizational and task context is unclear. How context could possibly be provided and how large groups of participating contributors can be handled may be learned from recent experiences in distributed software development. This is particularly interesting when taking place without existence of a formal organization as in the case of the distributed development of Linux and its components.

Certainly, more research is needed to understand the complex interdependence of using and tailoring systems collaboratively and to extend and enhance generalizable results for technical and organizational support for collaborative tailoring.

6. References

Note that where an URL is provided, it was last checked and valid in July 2001. The respective papers can be obtained from the author.

Avison, David; Lau, Francis; Myers, Michael; Nielsen, Peter Axel (1999): *Action Research*. In: Communications of the ACM, Vol. 42 (1). pp. 94-97.

Bannon, Liam; Schmidt, Kjeld (1991): *CSCW: Four Characters in Search of a Context*. In Bowers, John; Benford, Steve: Studies in Computer Supported Cooperative Work: Theory, Practice and Design. Amsterdam, North-Holland. pp. 3-16.

- Baskerville, Richard L.; Wood-Harper, A. Trevor (1996): *A Critical Perspective on Action Research as a Method for Information Systems Research*. In: Journal of Information Technology, Vol. 11 (3). pp. 235-246.
- Bentley, Richard; Dourish, Paul (1995): *Medium vs. Mechanism: Supporting Collaboration Through Customization*. In: Proceedings of ECSCW '95. Kluwer. pp. 133-148.
- Carstensen, Peter (1996): *Computer Supported Coordination*. Risø National Laboratory Risø-R-890(EN).
- Carter, Kathleen; Henderson, Austin (1990): *Tailoring Culture*. In: Reports on Computer Science and Mathematics no. 107, Åbo Akademi University 1990. Proceedings of 13th IRIS. pp. 103-116.
- Clarke, Anthony (1996): *A Theoretical Model of Cooperation*. In: Proceedings of COOP 1996. pp. 57-82.
- Dillenbourg, Pierre; Baker, Michael; Blaye, Agnes; O'Malley, Claire E. (1995): *The Evolution of Research on Collaborative Learning*. In: Reimann, Peter; Spada, Hans: Learning in human and machines. Towards an interdisciplinary learning science. London, Pergamon. pp. 189-211.
- Dourish, Paul; Bellotti, Victoria (1992): *Awareness and Coordination in Shared Workspaces*. In: Proceedings of CSCW '92. pp. 107-114.
- Engelskirchen, Torsten (2000): *Explorationsunterstützung in Groupware am Beispiel eines anpaßbaren Suchwerkzeugs* (English: Supporting Exploration in Groupware: the Example of a Tailorable Search Tool). Department of Computer Science III, University of Bonn. Diploma Thesis.
- Fischer, Gerhard; Girgensohn, Andreas (1990): *End-User Modifiability in Design Environments*. In: Proceedings of CHI '90. pp. 183-191.
- Galliers, Robert D. (1991): *Choosing appropriate information systems research approaches: a revised taxonomy*. In: Nissen, Hans-Erik; Klein, Heinz K.; Hirschheim, Rudy: The Information Systems Research Arena of the 90s, Challenges, Perceptions and Alternative Approaches. Amsterdam, North-Holland. pp. 155-173.

- Gantt, Michelle; Nardi, Bonnie A. (1992): *Gardeners and Gurus: Patterns of Cooperation among CAD Users*. In: Proceedings of CHI '92. pp. 107-117.
- Greenbaum, Joan; Kyng, Morten (1991): *Design at Work - Cooperative Design of Computer Systems*. Hillsdale, NJ, Lawrence Erlbaum Associates.
- Greenberg, Saul (1991): *Personizable groupware: Accommodating individual roles and group differences*. In: Proceedings of ECSCW '91. pp. 17-32.
- Grudin, Jonathan (1991): *Interactive Systems: Bridging the Gaps Between Developers and Users*. In: IEEE Computer, Vol. April 1991. pp. 59-69.
- Grudin, Jonathan (1994): *CSCW: History and Focus*. In: IEEE Computer, Vol. 27 (5). pp. 19-26.
- Gutwin, Carl (1997): *Workspace Awareness in Real-Time Distributed Groupware*. Department of Computer Science. Calgary, Alberta, University of Calgary. Ph.D. Thesis.
- Haaks, Detlef (1992): *Anpaßbare Informationssysteme - Auf dem Weg zu aufgaben- und benutzerorientierter Systemgestaltung und Funktionalität*. Göttingen und Stuttgart, Verlag für Angewandte Psychologie.
- Henderson, Austin (1997): *Tailoring Mechanisms in Three Research Technologies*. In Workshop on Tailorable Groupware: Issues, Methods, and Architectures at the Group '97 organized by Mørch, Anders; Stiernerling, Oliver; Wulf, Volker. Phoenix, Arizona, USA. <http://www.ifi.uib.no/staff/anders/research/group97/wspapers/henderson.ps>.
- Henderson, Austin; Kyng, Morten (1991): *There's No Place Like Home: Continuing Design in Use*. In Greenbaum, Joan; Kyng, Morten: *Design at Work - Cooperative Design of Computer Systems*. Hillsdale, NJ, Lawrence Erlbaum Associates. pp. 219-240.
- Herrmann, Thomas; Wulf, Volker; Hartmann, Anja (1996): *Requirements for a Human-centered Design of Groupware*. In Shapiro, Dan; Tauber, Michael; Traunmüller, Roland: *Design of Computer Supported*

- Cooperative Work and Groupware Systems. Amsterdam, Elsevier. pp. 77-99.
- Henry, Paul David (1997): *PALIMPSEST - a model for Networked Hypermedia and Distance Learning*. <http://www.programhouse.com/pal/paltext.htm>.
- JavaSoft (1997): *JAVABEANS 1.0 API Specification*. Mountain View, California, SUN Microsystems.
- JCSCW - Computer Supported Cooperative Work (2000): *The Journal of Collaborative Computing*. Vol. 9 (1). Special Issue on Tailorable Systems and Cooperative Work.
- Kahler, Helge; Rittenbruch, Markus (1999): *Structuring Information in a Virtual Organization - A Web-based Approach*. In Bullinger, Hans-Jörg; Vossen, Paul Hubert: Adjunct Conference Proceedings of the HCI International '99. pp. 278-279.
- Kjær, Arne; Madsen, Kim Halskov (1994): *Participatory Analysis of Flexibility: Some Experiences*. In: Proceedings of Participatory Design Conference '94. pp. 21-31.
- Klöckner, Konrad; Mambrey, Peter; Sohlenkamp, Markus; Prinz, Wolfgang; Fuchs, Ludwin; Kolvenbach, Sabine; Pankoke-Babatz, Uta; Syri, Anja (1995): *POLITeam: Bridging the Gap between Bonn and Berlin for and with the Users*. In: Proceedings of ECSCW '95. pp. 17-31.
- Kühme, Thomas; Dieterich, Hartmut; Malinowski, Uwe; Schneider-Hufschmidt, Matthias (1992): *Approaches to Adaptivity in User Interface Technology: Survey and Taxonomy*. In: Proceedings of IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction 1992. Elsevier, North-Holland. pp. 225-252.
- Lemken, Birgit; Kahler, Helge; Rittenbruch, Markus (2000): *Sustained Knowledge Management by Organizational Culture*. In: Proceedings of 33rd Hawaii International Conference on System Sciences (HICSS-33).
- Mackay, Wendy E. (1990): *Patterns of Sharing Customizable Software*. In: Proceedings of CSCW '90. pp. 209-221.
- Mackay, Wendy E. (1991): *Triggers and Barriers to Customizing Software*. In: Proceedings of CHI '91. pp. 153-160.

- MacLean, Allan; Carter, Kathleen; Lövstrand, Lennart; Moran, Thomas (1990): *User-Tailorable Systems: Pressing the Issues with Buttons*. In: Proceedings of CHI 90. pp. 175-182.
- Malone, Thomas W.; Grant, Kenneth R.; Lai, Kum-Yew; Rao, Ramana; Rosenblitt, David (1988): *Semistructured Messages are Surprisingly Useful for Computer-Supported Coordination*. In: Proceedings of CSCW 88. Morgan-Kaufmann Publishers. pp. 311-334.
- Malone, Thomas W.; Lai, Kum-Yew; Fry, Christopher (1992): *Experiments with Oval: A Radically Tailorable Tool for Cooperative Work*. In: Proceedings of CSCW '92. pp. 289-297.
- Mambrey, Peter; Mark, Gloria; Pankoke-Babatz, Uta (1996): *Integrating user advocacy into participatory design: The designers' perspective*. In: Proceedings of PDC '96. pp. 251-259.
- Mørch, Anders (1995a): *Three Levels of End-user Tailoring: Customization, Integration, and Extension*. In: Computers in Context: Joining Forces in Design. Aarhus. pp. 157-166.
- Mørch, Anders (1995b): *Application Units: Basic Building Blocks of Tailorable Applications*. In: Proceedings of East-West Int'l. Conference on HCI. Springer. pp. 68-87.
- Nardi, Bonnie A.; Miller, James R. (1991): *Twinkling lights and nested loops: distributed problem solving and spreadsheet development*. In: Int. J. Man-Machine Studies, Vol. 34. pp. 161-184.
- Nielsen, Morten; Carstensen, Peter (1998): *Cooperation and Interdependence*. In: Proceedings of 21st Information Systems Research seminar in Scandinavia (IRIS). Department of Computer Science, Aalborg University, Denmark. pp. 657-676.
- Oberquelle, Horst (1994): *Situationsbedingte und benutzerorientierte Anpaßbarkeit von Groupware*. In Hartmann, Anja; Herrmann, Thomas; Rohde, Markus; Wulf, Volker: *Menschengerechte Groupware - Software-ergonomische Gestaltung und partizipative Umsetzung*. Stuttgart, Teubner. pp. 31-50.
- Oppermann, Reinhard (1989): *Individualisierte Systemnutzung*. In: Proceedings of 19. Jahrestagung der GI. Springer. pp. 131-145.

- Oppermann, Reinhard, Ed. (1994): *Adaptive User Support: Ergonomic Design of Manually and Automatically Adaptable Software*. Hillsdale, New Jersey, Lawrence Erlbaum Associates.
- Oppermann, Reinhard; Reiterer, Harald (1992): *Das Konzept der Anpassung als Voraussetzung für eine Individualisierte und kooperative Nutzung von EDV-Systemen*. GMD Arbeitspapiere 625.
- Pipek, Volkmar; Wulf, Volker (1999): *A Groupware's Life*. In: Proceedings of ECSCW '99. Kluwer. pp. 199-218.
- Rittenbruch, Markus; Kahler, Helge; Cremers, Armin B. (1998): *Supporting Cooperation in a Virtual Organization*. In: Proceedings of ICIS '98. pp. 30-38.
- Robinson, Mike (1993): *Design for unanticipated use.....*. In: Proceedings of ECSCW '93. Kluwer. pp. 187-202.
- Schmidt, Kjeld (1994): *Modes and Mechanisms of Interaction in Cooperative Work. Outline of a Conceptual Framework*. Risø National Laboratory Risø-R-666(EN).
- Sørensen, Carsten (1994): *This is Not an Article — Just Some Thoughts on How to Write One*. In: Proceedings of 17th Information Systems Research seminar in Scandinavia (IRIS). pp. 46-59.
- Stallman, Richard (1981): *EMACS: The Extensible, Customizable, Self-Documenting Display Editor*. In: Proceedings of ACM SIGPLAN SIGOA. Massachusetts Institute of Technology. pp. 301-323.
- Trigg, Randall H. (1992): *Participatory Design meets the MOP. Accountability in the design of tailorable computer systems*. In: Proceedings of 15th Information Systems Research seminar in Scandinavia (IRIS). pp. 643-656.
- Trigg, Randall; Bødker, Susanne (1994): *From Implementation to Design: Tailoring and the Emergence of Systematization in CSCW*. In: Proceedings of CSCW '94. pp. 45-54.
- Trigg, Randall H.; Moran, Thomas; Halasz, Frank (1987): *Adaptability and Tailorability in NoteCards*. In: Proceedings of Human-Computer Interaction-Interact'87. pp. 723-728.
- Wasserschaff, Markus; Bentley, Richard (1997): *Supporting Cooperation through Customisation: The Tviews Approach*. In: Computer

- Supported Cooperative Work: The Journal of Collaborative Computing (JCSCW), Vol. 6. pp. 305-325.
- Woods, David D. (1988): *Coping with complexity: The psychology of human behavior in complex systems*. In Goodstein, Len P.; Andersen, Henning Boje; Olsen, Svend Erik: *Mental Models, Tasks and Errors*. London, Taylor & Francis. pp. 128-148.
- Wulf, Volker; Stiernerling, Oliver; Pfeifer, Andreas (1999): *Tailoring Groupware for Different Scopes of Validity*. In: *Behaviour & Information Technology*, Vol. 18 (no 3). pp. 199-212.

Second Part of Thesis

First Paper

From Taylorism to Tailorability

Supporting organizations with tailorable software and object orientation

Abstract

With markets globalizing and customers' demands specializing organizations worldwide need to change. To reach the necessary flexibility of information technology one approach is tailorability, i.e. users are enabled to adjust software to their needs. Some examples for tailorability are given, and its potential benefits and shortcomings are discussed. Software development plays an important role for establishing tailorability, and object oriented methods can be helpful in this context.

Intro

The former successful tayloristic work model that divided labor to increase efficiency is now a big obstacle for modern business with its need to react quickly to environmental changes. The conviction that there is one best way for an organization to run and that organizations work like machines which follow the linear principles of cause and result is replaced by the idea of an organization as a constantly moving social network that keeps adapting to environmental changes (Paetau 1994). Some of the newer literature on organization introduced new concepts to overcome the rigidity of taylorism and paid tribute to a less linear understanding of organizations. The suggestions include "reengineering the corporation" (Hammer & Champy 1994) and building "virtual" (Davidow & Malone 1993), "fractal" (Warnecke 1993), or "object oriented" (Klotz 1993) organizations. All of them differ in their main focus, but all stress the importance of information technology while skipping its particular role and shape in a post-tayloristic setting.

That this must not be neglected in any organization we learned in a project finished recently in which we developed software ergonomical design principles for groupware (cf. Herrmann et al. 1993). The principles were influenced by interviews we conducted and that supported our hypothesis that the usage of networks which are usually designed for communication and co-operation also raises conflicts of interests between different users. An example for this is the ambiguity of visibility in a network of cooperating people: Whereas it can be of importance to one person to see what other people in a team work on it might be felt to be unwanted control by the latter. In a situation like this no single optimal solution will be available. Instead a common stable solution for all participants must be found. Moreover, the system should be able to handle ad-hoc-negotiations about conflicts by means of a negotiation function (cf. Wulf 1993).

In order to reach the organizational flexibility demanded by newer organizational theory as well as by the software ergonomical need to adjust systems to individuals and groups several measures and concepts are promising. One of them is *tailorability*.

Tailorability

Software for a modern organization should be tailorable. This means that users can adjust it to their special needs by themselves. Ideally there will be different levels of adjustment for different needs and qualifications (cf. Henderson & Kyng 1991). Thus, adjusting the software might mean that a person places icons or a toolbar on the screen wherever she or he wants, that the input device can be chosen (keyboard, mouse, voice etc.), but also that people make highly sophisticated configurations in a system to support their work as a team e. g. by defining who's substituting whom at which occasion in a workflow process. Also end user programming can be used to tailor software.

Relevant dimensions of the tailoring process are who the initiator, the actor, and the affected persons are, what its subject is (user, task, organizational context), what its goal is, or when, for how long, and for what parts of the system it is made (cf. Haaks 1992).

Some examples of collaborative work practice can highlight different aspects of tailoring software in a collaborative work setting.

- When introducing a new ISDN telephony system in an organization several configurations can be made. Some of them concern technical aspects, others may result from the organizational structure and again others relate to privacy aspects. Thus, one part of the configuration adopts the telephony software to the standards of the local telephone company while another part sets the rules for call forwarding or prevents any unauthorized person to activate the microphone of one of the telephones from a remote place. Usually these configurations affect all of the people using the telephone system, and they are rarely changed. They are made *before the first use of the system*. The telephone users are seldom asked to participate in the configuration process or even informed. This is a classical and rather rigid form of tailoring.
- With SHARE Greenberg (1991) proposes a layered architecture for a conference tool with the possibility to use one of a set of “personizable floor control policies”, i.e. rules for deciding who’s turn it is next to speak/write in a computer mediated conference. Such a rule could state that every person that wants to say something can just “grab” the turn by interrupting the speaking person, or that the speaking person has to explicitly stop speaking before anybody else can start. Also, there could be a chairperson who picks the next speaker from the group who raise their hands (shown by an icon on the screen). Other floor control policies can be defined. The decision for a special policy is made *before every single session*. It remains open if only the person initiating a meeting or the whole group can decide about a meeting’s floor control policy.
- A closer look at organizational aspects of tailoring is taken in some papers dealing with the sharing of customization files (Mackay 1990; Nardi 1993 Chapter 6; Trigg & Bødker 1994). While the software that is dealt with is not necessarily groupware, i.e. used by people to collaborate, there are still interesting observations about collaborative work practice made. The main point of interest here is that people individualize software they use for their daily work and share these individualized custom files with others who feel that the files are useful for them. The customization may include some macros or lengthy blocks of standard text that different people need to write to fulfill legal or other requirements. To have others benefit from the customization files “translators” (Mackay 1990) are needed who make custom files accessible and talk to people about them, thus “translating” between the

system developers and the end users. They should be domain experts as well as interested in computers and willing and able to communicate and help people. The translators often emerge from a group in a “natural” way. They can be beneficial for an organization in supporting the process of tailoring software to an organization’s needs. Some positive experiences with translators (“gardeners”, as Nardi (1993) calls people who are rewarded or paid for being translators) have been made. They have shown that gardeners can be a source of high quality support. Other field studies have mentioned that the process of sharing custom files has become more systematized in the course of the time (Trigg & Bødker 1994).

- Malone et al. (1992) describe a system that allows end users to tailor software on a level closer to system development than just setting parameters. Their OVAL System is a “radically tailorable tool for cooperative work” where “radically tailorable” means that the tool is meant to enable end users to create different applications by modifying a working system. This is done by combining and modifying *objects*, *views*, *agents*, and *links* which provide an elementary tailoring language. While, in fact, the idea of end users designing the application that suits them best is intriguing, the question remains how many users will be able to handle the more advanced features of this complex system.

A very important aspect of tailoring in a work group setting has not yet been widely discussed. None of the examples mentioned involved a group of users jointly tailoring their groupware system to the group’s needs. This will be of increasing importance in the future since more and more groupware systems will be installed in post-tayloristic organizations. Having a group tailor their groupware system raises questions about how this can be done. Similar to the technical (cf. Wulf 1995) and organizational means to deal with conflicts among groupware users while they use the system, ways have to be found to moderate different interests in a groupware tailoring process.

The tailoring examples above highlight some of the potential benefits and shortcomings of tailoring software. Generally tailorable software can enhance the flexibility of an organization by enabling technical adjustments to organizational needs. Thus, for an organization tailoring can be a vital part of the management of change and the heading towards a learning organization. By actively supporting the tailoring process e.g. by gardeners spreading custom files or by work groups discussing group-relevant aspects

of tailoring the self-organizational potential of work groups in a post-tayloristic organization can be set free. Also organizational support by gardeners or regular tailoring meetings can be helpful to provide structure to the tailoring process and thus keeping the organization's technical infrastructure from turning into a thicket of incompatible individual solutions or being lost in space between system planets tailored by groups.

On the other hand tailoring software is not inevitably beneficial. There is the danger of a tailoring overhead since the time and effort needed for tailoring is lost - at least at a first glance - for the primary work task. Gardeners might be difficult to find, since they need a double qualification finding themselves on the border of system development and everyday work. Also, there is a danger of getting too unfamiliar with the work settings when a gardener works for a longer time on the developer's side of the border. Moreover, the gardener system provides another layer between system developers and end users. While this is intended to improve the communication between these groups it might just increase the organization's hierarchical overhead. Organizations face the danger of running into a qualification problem not only for gardeners but for all personnel. Tailoring software requires technical and social skills, particularly in a collaborative work environment, that not every organization member might have or be willing or able to acquire. Moreover, software ergonomical design principles particularly for groupware and work psychological demands are in danger of being disrespected by unskilled tailors. This is a dilemma software ergonomics faces by pleading for tailorability to ensure that a system fits the users' needs on one side while on the other side struggling with the potential negative effects of letting the final look and functionality of a system slip out of the hands of developers and putting it into the responsibility of people tailoring within an organization. Finally, allowing for more internal dynamics as a result of continuously tailoring software might be a problem for some managers fearing to lose control.

Deciding on how a system needs to be to fulfill the expectations of all or most of the people working with it at least to a certain extent doesn't get easier when the decision is made within an organization rather than by external system developers. But the chances are better to find a way of tailoring the system to support the organization and its members well by discussing the special needs and trying out what works good and what doesn't.

Software Development

Tailorability is a feature of software. Therefore, software development is a relevant area to look at. Moreover, software development is affected by the post-tayloristic organization models directly. Developing software for a special organization must no longer be document driven and follow the rigid top-down waterfall model, i.e. consist of predefined disjunct phases with written milestones to document the state of a project before the next phase may be started.

To overcome the waterfall model several approaches have been made. Two of the most promising are EOS (Evolutionary Object Oriented System Development) (Hesse & Wertz 1994) and STEPS (Software Technology for Evolutionary Participative System Development) (for an introduction cf. Floyd et al. 1989). Both consider software development to be evolutionary and stress the necessity to continuously coordinate the areas of development and usage since system development always includes the design of workplaces.

The particular strength of EOS is the usage of object orientation (see chapter below) which might be a good *technical* basis to realize tailorable systems. STEPS relies on user participation where developers and users exchange their views of the system-to-be or the last revision to ensure that the software fits the users' current and future needs. Thus, part of the tailoring before the continuous usage of the software is done by system developers together with end users. This procedure for the two groups to work together in evolutionary development seems to be a good *social* approach to tailoring.

Remarks on Object Orientation

There are different relations between the computers science's concept of object orientation and post-tayloristic organizations. Klotz (1993) has used the object oriented image of independently operating objects communicating through clearly defined interfaces and being provided with all necessary resources to describe how a modern organization should work and called it "object oriented organization". It is yet an open question if this is not overstressing the comparison with the object oriented approach.

But still object orientation can be of great value in designing and using tailorable software. Since people tend to think in objects rather than functions the interaction in a participatory design process between developers and users on what the work and the software are about is made a lot easier with the gap between their different notions narrowing. On the other hand object orientation supports an evolutionary process by allowing easy prototyping and changing of modules.

Basic to object orientation are modularity by data encapsulation, inheritance of object features, and the polymorphism that frees objects that send a message from knowing the receiving object's properties. Each of them can be very helpful for tailorability, e. g.:

- encapsulation ensures lean and clearly defined interfaces, so parts of the software can relatively easy be changed, removed, or added without risking a decrease of system stability;
- inheritance particularly supports working in a group where features of a configuration object of the work group can be passed on to the group members' configuration objects without them having to configure each and every of their work environments manually whenever the group configuration changes;
- polymorphism is needed in tailored work settings to ensure that changing one object doesn't make it necessary to change the methods of other objects.

These features have shown to be valuable for prototyping and will be of use to build tailorable software since many of the requirements for prototyping and tailorability are common. Moreover, for tailorable software a layered architecture seems to be useful, where each layer is responsible for one aspect of tailorability, e. g. the user interface or building and using macros. To keep these layers independent and stable to the frequent changes resulting from tailoring object orientation can provide the methods.

Extro

Tailorability can be part of the solution to the problems that organizations face in a fast changing market. On the other hand it imposes more work on an organization since it takes part of the responsibility for the adequate realization of the software from the system developers. After all, building

tailorable software and working with it is a process of deregulation with the chances and dangers depending on an organization's ability to use the potential of tailorability.

More work must be directed on the tailoring practice for different settings, including the development of gardening models, concepts for tailoring as a group process, and the evaluation of tailoring activities. Moreover, technical advancement towards architectures for tailorable software must be made. Object oriented concepts might be helpful here. Finally, it is crucial to understand that organizational and technical flexibility and change are intertwined and must be dealt with jointly (cf. Rohde & Wulf (1995) for suggestions for an integrated organization and technology development).

References

- Davidow, William H.; Malone, Michael S. (1993): *Das virtuelle Unternehmen. Der Kunde als Koproduzent*. Frankfurt am Main und New York, Campus.
- Floyd, Christiane; Reisin, Fanny-Michaela; Schmidt, Gerhard (1989): *STEPS to Software Development with Users*. In Ghezzi, Carlo; McDermid, John A.: ESEC'89 - 2nd European Software Engineering Conference, University of Warwick, Coventry. Heidelberg, Springer. pp. 48-64.
- Greenberg, Saul (1991): *Personizable groupware: Accommodating individual roles and group differences*. In: Proceedings of ECSCW '91. pp. 17-32.
- Haaks, Detlef (1992): *Anpaßbare Informationssysteme - Auf dem Weg zu aufgaben- und benutzerorientierter Systemgestaltung und Funktionalität*. Göttingen und Stuttgart, Verlag für Angewandte Psychologie.
- Hammer, Michael; Champy, James (1994): *Reengineering the Corporation - a Manifesto for Business Revolution*. New York, Harper Business.
- Henderson, Austin; Kyng, Morten (1991): *There's No Place Like Home: Continuing Design in Use*. In Greenbaum, Joan; Kyng, Morten: Design at Work - Cooperative Design of Computer Systems. Hillsdale, NJ, Lawrence Erlbaum Associates. pp. 219-240.

- Herrmann, Thomas; Wulf, Volker; Hartmann, Anja (1996): *Requirements for a Human-centered Design of Groupware*. In Shapiro, Dan; Tauber, Michael; Traunmüller, Roland: *Design of Computer Supported Cooperative Work and Groupware Systems*. Amsterdam, Elsevier. pp. 77-99.
- Hesse, Wolfgang; Weltz, Friedrich (1994): *Projektmanagement für evolutionäre Software-Entwicklung*. In: *Information Management*, Vol. 1994 (3). pp. 20-32.
- Klotz, Ulrich (1993): *Vom Taylorismus zur Objektorientierung*. In Scharfenberg, Heinz: *Strukturwandel in Management und Organisation*. Baden-Baden, FBO-Verlag. pp. 161-199.
- Mackay, Wendy E. (1990): *Patterns of Sharing Customizable Software*. In: *Proceedings of CSCW '90*. pp. 209-221.
- Malone, Thomas W.; Lai, Kum-Yew; Fry, Christopher (1992): *Experiments with Oval: A Radically Tailorable Tool for Cooperative Work*. In: *Proceedings of CSCW '92*. pp. 289-297.
- Nardi, Bonnie M. (1993): *A Small Matter of Programming*. Cambridge, Massachusetts, MIT Press.
- Paetau, Michael (1994): *Configurative Technology: Adaption to Social Systems Dynamism*. In Oppermann, Reinhard: *Adaptive User Support - Ergonomic Design of Manually and Automatically Adaptable Software*. Hillsdale, N.J., Lawrence Erlbaum. pp. 194-234.
- Rohde, Markus; Wulf, Volker (1995): *Introducing a Telecooperative CAD-System - The concept of Integrated Organization and Technology Development*. In: *Proceedings of International Conference on Human Computer Interaction (HCI) '95*. pp. 787-792.
- Trigg, Randall; Bødker, Susanne (1994): *From Implementation to Design: Tailoring and the Emergence of Systematization in CSCW*. In: *Proceedings of CSCW '94*. pp. 45-54.
- Warnecke, Hans-Jürgen (1993): *Revolution der Unternehmenskultur*. Berlin, Springer.
- Wulf, Volker (1993): *Negotiability: A Metafunction to Support Personable Groupware*. In: *Proceedings of International Conference on Human Computer Interaction (HCI) '93*. pp. 985-990.

Wulf, Volker (1995): *Mechanisms for Conflict Management in Groupware*.
In: Proceedings of International Conference on Human Computer
Interaction (HCI) '95. pp. 379-385.

Second Paper

How to Make Software Softer - Designing Tailorable Applications

Abstract

The design of tailorable systems is an important issue for fields of application which are characterized by differentiation and dynamics. We show how tailorability can be combined with approaches of evolutionary and participative software engineering and discuss some conceptual problems arising from this approach. Moreover, we present two case studies on how to design tailorable functionality in a groupware development project.

Introduction

Tailorability is a property of software which allows to change certain aspects of the software in order to meet different user requirements. It is widely agreed that tailorability is one of the major future challenges in the design of user interfaces and interactive systems (Trigg 1992, Carter & Henderson 1990, MacLean et al. 1990, Henderson & Kyng 1991, Olsen et al. 1993, Malone et al. 1992, Kahler 1995, Bentley & Dourish 1995).

Several authors have pointed out that with tailorable software one has to take into account several problems which classical design methodologies do not (and do not have to) address. On a technical level, the software architecture has to provide means of changing system behavior other than rewriting and recompiling source code. Henderson & Kyng (1991), Haaks (1992), Mørch (1997) stress the basic flexibility of object oriented architectures in this respect. For instance, OVAL (see Malone et al. 1992) is based on four elements (objects, views, agents and links) which constitute a language which can be used to rapidly build and tailor groupware

applications. LINKWORKS by DEC (see DEC 1995) is another example of a tailorable system based on an object oriented architecture. The system provides a set of high level language elements and tools for deriving new classes of application objects and redefining system behavior. Tailorability as understood by the designers of the systems mentioned above goes very far, allowing to build - from the same construction set - full fledged applications which may serve rather diverse purposes. The basic complexity of creating an application, however, steers this brand of tailorability towards the community of professional designers, resulting in powerful and efficient high-level design-tools for building and maintaining software.

On a more user centered level, tailorability can be regarded as the means to adapt existing applications to changes in the needs of single users or groups of users, making the software better fit the current work situation. Examples are the recording of macros in word processors to automate sequentially executed tasks, the implementation of an access policy using mechanisms for discretionary access control or just changing the screen to the current user's favorite color. The basic complexity of these actions is not beyond the scope of end users (see Nardi & Miller 1991, Henderson & Kyng 1991). Tailorability of this kind, however, provides several new challenges for the design process of software.

In this paper we will focus on the design process for tailorable software. We will present two rather different cases studies out of the context of the POLITEAM Project (see Klöckner et al. 1995) in order to show how end-user tailorability can be accommodated in a participative design approach. The first section examines the initial motivation for making software tailorable and from this perspective derives a number of questions which have to be addressed during the design process. We then focus on the task of capturing diversified and dynamic requirements. The second section gives a short overview over the context provided by the POLITEAM Project. In the third section we discuss two actual design cases in this context: the redesign of a search tool for documents and the redesign of the discretionary (i.e. user tailorable) access control system in a groupware system. The conclusion sums up the lessons learnt from these experiences and presents questions which have not yet been answered.

Designing Tailorable Software

In this section, we want to examine the initial motivation for designing tailorable software in order to derive and clarify the questions which have to be addressed in the design of tailorable software.

Traditional software design following the waterfall model (see Boehm 1976) is concerned with capturing, realizing and testing one set of requirements, reflecting a snapshot of one field of application (see figure 1). The field of application may be a specific organization with special requirements concerning functionality and interface features.

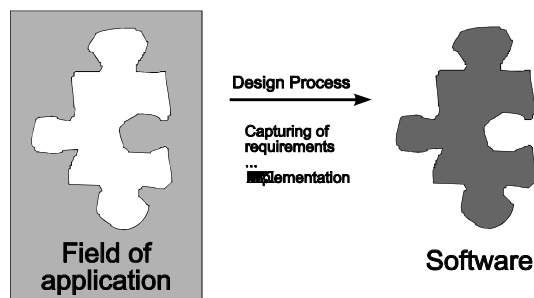


Figure 1: Designing software which meets the requirements of one field of application at one time

Software development according to the waterfall model focuses on one field of application and assumes that the requirements for system design are clear at the very beginning of a project and stable for a long period of time. Both of these underlying assumptions have been questioned for years. Therefore, evolutionary approaches to software engineering try to capture dynamically evolving requirements employing an iterative design procedure (see Boehm 1985, Henderson-Sellers & Edwards 1990). In participative and evolutionary approaches the users of an application are actively involved in the design process and are thus given the opportunity to articulate their requirements (see Floyd et al. 1989, Grønbæk et al. 1995).

These approaches indicate a viable way to overcome problems of traditional software development methods. Nevertheless they require the involvement of the system developers whenever a new requirement is pointed out by the users. In working environments which are characterized by a high degree of dynamics in user requirements, the continuous involvement of the system

developers may retard or even impede a necessary adaptation of the software. Therefore, Wulf & Rohde (1995) have proposed to combine evolutionary and participative software development with activities of tailoring in use. Since tailoring can be performed by users, local experts, or support staff, the implementation time for small changes may be reduced significantly which often appears to be critical to the success of an application.

Figure 2 shows how tailoring activities can be combined with evolutionary and participative software-development. This combination extends the STEPS process model developed by Floyd et al. (1989).

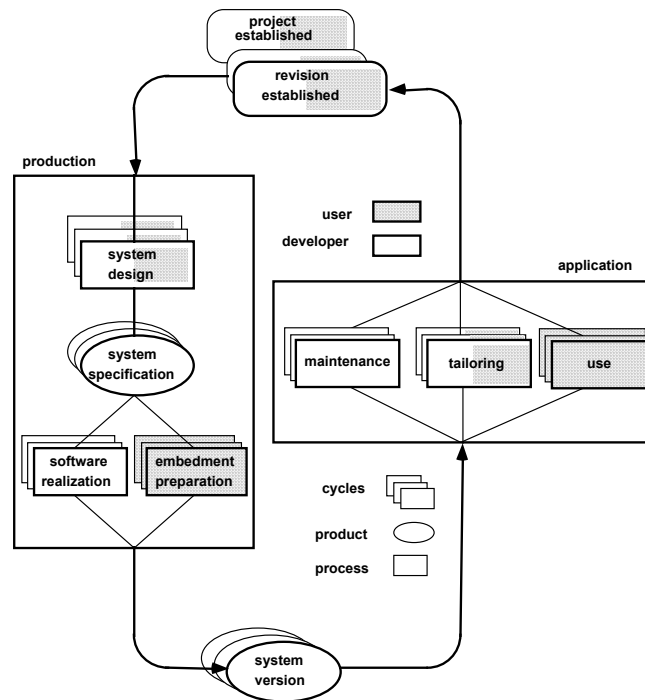


Figure 2: Extended STEPS-approach - Combining evolutionary and participative software development with tailoring in use (see Wulf & Rohde 1995)

Apart from dynamically evolving user requirements, diversity of requirements is another reason for designing tailorable software. Requirement diversity is encountered, for instance, in product development for large markets. Tailorability allows a generic product to satisfy the diverse demands of many customers. Additionally, diversity is encountered

in the development of custom-made multi-user software, especially groupware, as inter-individual differences as well as different organizational roles or tasks may require distinct views on data or different functionality.

Thus, dynamically evolving and differentiated requirements from different fields of application are the main reason for the development of tailorable software (see Henderson & Kyng 1991, Trigg et al. 1987, Paetau 1994, Oberquelle 1994). Taking these considerations into account, the design process for tailorable applications has to capture the diversity and dynamics of the fields of application, as shown in figure 3.

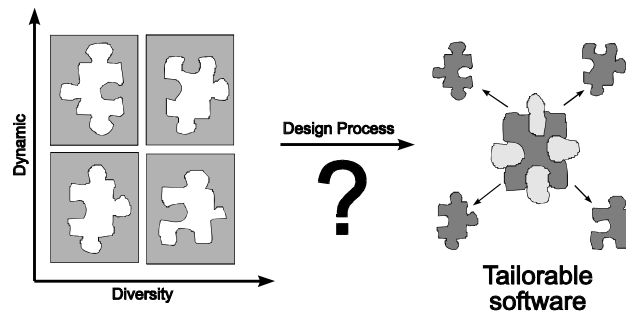


Figure 3: The central question of this paper: how to design tailorable software?

Considering this modified view of designing, we have to deal with the following questions concerning the particularities of designing flexible software:

- How can the designer capture diversified and future requirements and how can he distill the necessary *range of flexibility* from these requirements?
- How can this range of flexibility be implemented technically, leading to the question of *software architecture*?
- How can the technical flexibility offered by the architecture can be made accessible for end users through the *user interface*?

In the rest of the paper we focus primarily on the first question in the context of experiences from the POLITEAM project. In the next subsection, we want to take a closer look at the relationship between tailorability and evolutionary & participatory design.

The Relationship of Tailorability and Evolutionary & Participatory Design

Kjær and Madsen (1994) have applied participatory techniques in analyzing the requirements for flexibility of a picture archive and communication system in a hospital. They conclude (p. 22):

“... flexibility concerns not the regular procedures and standard way of doing things but the unexpected, unprecedented, the exceptional cases, situations and events which are only experienced by the people who do the day to day work.”

Thus, participatory & evolutionary design can - on one hand - be employed to design the “right kind” of tailorability. In this sense, it is used to expose diversity in requirements in one or several fields of application.

On the other hand, tailorability and participatory & evolutionary design are complementary as discussed in the last section. In this other sense, the purpose of tailorability is to make the software more robust to small anticipated changes and diversities in requirements in-between phases of design, while the evolutionary redesign aims at taking into account more significant and unexpected changes.

For the purpose of the rest of this paper we want the relationship between tailorability and participatory & evolutionary design to be understood in the first sense, i.e. capturing diversified (and - in a limited sense - future) requirements.

Capturing Diverse and Future Requirements

To determine the range of flexibility to be supported by a tailorable application, two (at first glance) different problems have to be addressed. On one hand, the analysis technique has to capture the diversity of existing requirements in different fields of application across a market segment or across different subfields of application in the same organization. On the other hand, it has to “predict” future requirements.

How can the designer go about addressing the first problem, the capturing of diversity between or within organizations and persons?

Assuming that we cannot involve every possible user in our design process (a safe assumption in large organizations and markets), the first obvious step is a careful selection of users. The selection has to be careful in the sense that we still want to capture the full range of requirements in order to make the system as flexible as necessary. The importance of a careful selection is illustrated by an example reported by Grudin (1989). He describes the unsuccessful development of a group scheduling tool, whose failure was caused by only including managers into the analysis stage of the design process. As was discovered later the subordinates had rather different requirements which lead to a lethally low acceptance rate of the final product.

The selection process at this stage must necessarily be of an explorative and heuristic nature. In our design cases described later on, we developed such heuristic selection schemes. The selection criteria for users and organizations in these schemes were based on our past experiences concerning e.g. the differences of access policies encountered in public and private organizations and - admittedly - pure guesswork. We also had to take into account practical limitations of accessibility and time in selecting participants for the analysis stage of the design processes.

As the designers learn more about the diversity of requirements in relation to different users and organizations, the selection scheme should be refined to accommodate newly discovered correlations between requirements and user or organization types. This refinement may be supported by including users and organizations into the scheme, which are - according to the current version of the selection scheme - redundant. If, for example, the scheme calls for a distinction of users in private and public organizations, redundancy may be achieved by including several private, respectively public organizations in the analysis stage. Thus, if the scheme does not capture all dependencies between user (or organization) types and requirements, some supposedly redundant entities are prone to exhibit diverse requirements, as well. This redundancy allows for a small degree of fault-tolerance in the scheme.

We also suggest to actively investigate the cause for the individual requirements in each case as another way to refine the heuristic scheme on the fly. The goal must be to find a correlation between certain attributes of the examined entities (organizations or persons) and the requirement encountered, e.g. organizations with a high degree of exposure to market

pressures exhibit the need for time-dependent access policies (see our case study).

We are aware that our heuristic approach cannot guarantee correct results, but we believe that it can serve as an efficient tool to obtain at least a rough idea about the degree of flexibility necessitated by the diversity of requirements.

But what about future requirements? Ecklund et al. (1996) suggest to extend Jacobsen's *use-case* methodology (see Jacobsen et al. 1992) with the concept of *change-cases* in order to accommodate future changes in requirements. Their approach, however, concentrates on the expression of possible changes within the use-case methodology and the tracing of changes to other levels in the design process. Concerning the capturing of future requirements they only suggest to take into account (p. 354):

- *“planned or scheduled changes to product / services offerings*
- *user comments [...]*
- *review of regulatory / legal environment*
- *drafts of pending legislation / regulations*
- *review of organization's technology & platform strategy.”*

These points definitely are important and have to be taken into account during design. They are useful to accommodate clearly specifiable changes, for example, in the rate of value added tax in accounting systems. We believe, however, that there are changes in the environment, the consequences of which on the software cannot be easily specified. Regard, for example, a small but fast-growing company which wants to introduce a new (custom-made) email system. The company plans to multiply its workforce in the next few years. The consequences concerning the number of possible email accounts are easily deducted from the expected growth. But what about the way people use email? Will email-filters become more important, will people write less emails “to all users”, etc. These questions are not easily answered.

We suggest to extend our heuristic selection scheme to take into account the factors which drive organizational change (e.g. growth, learning, change of environment). This brings us back to the necessity of carefully selecting

users and organizations for participation. One could explicitly select organizations (or individuals) which are further along an assumed “change-curve” (e.g. bigger companies, companies with a more dynamic environment, more experienced users, etc.) in order to gain clues concerning future requirements and usage patterns.

However, one has to keep in mind, that there are some factors driving organizational and individual change (e.g. new technologies) which prevent finding example organizations or persons which represent possible “futures”.

In the rest of the paper we describe and discuss how we have employed the thoughts and concepts presented in this section in two design cases.

The POLiTEAM Project

Our design cases are taken from the context of the POLiTEAM project. Within the POLiTEAM project a groupware application for a German federal ministry and selected ministries of a state government and the concurrent engineering division of a car producer is developed in an evolutionary and participative way. The first system version was generated by configuring the commercial product LINKWORKS by Digital. Based on the experiences gained by introducing the first system version in three different fields of application, we develop advanced versions of the system. The functionality mainly consists of an electronic circulation folder, shared workspaces, and an event notification service.

The project started by carrying out semi-structured interviews with future users in the three fields of application to learn about their work practice. This information was used to generate a prototypical configuration of the commercial product for each of the different fields of application. This prototype was presented in a workshop, modified accordingly and finally introduced as the POLiTEAM I system. After the introduction the users were supported regularly by project members who communicated their experiences to the designers of the next system version (user advocates, see Mambrey et al. 1996). Moreover, interviews were carried out and workshops were held regularly to allow for direct communication between designers, support staff and users.

One major problem with the first versions of POLiTEAM was the insufficient protection of privacy. Since the system is based on a desktop metaphor, the

users expected the system to regulate visibility and accessibility of documents according to this metaphor. Unfortunately, the protection mechanisms did not conform with this requirement, as they were completely independent from the virtual desktops and folders. Users could access any object - if not explicitly denied by an access profile - on any desktop across the whole virtual office using a search tool (The result of using the search tool was a set of links through which all found objects could be accessed). The users were very suspicious about which aspects of their work were open to inspection by superiors and colleagues, because the specification of access rights was very ambiguous and unclear (see second case study).

A first-cut solution was the removal of the search tool, limiting the initiation of cooperation to explicitly sending links to shared workspaces to all cooperators. As this was not a very satisfying solution to the problem, the POLITEAM group at the University of Bonn decided to make privacy a central theme in their redesign efforts. Since the search tool and the access rights were clearly identified as the major culprits of the suboptimal design, we decided to pursue two complementary venues of redesigning the product. One subgroup concentrated on redesigning the search tool, while the other took on the redesign of the access control system. The two groups met regularly to coordinate their designs and produce an integrated solution. In the following section we describe our experiences during the two design processes.

Two design cases

Design of an Access Control System for Groupware

The need for making the discretionary access control system more flexible was - as mentioned above - one of the premiere requirements voiced by users during the early stages of the participatory design process. In this section we describe the steps of this process concerning the design of a new access control system. Figure 5 shows the basic structure of this process.

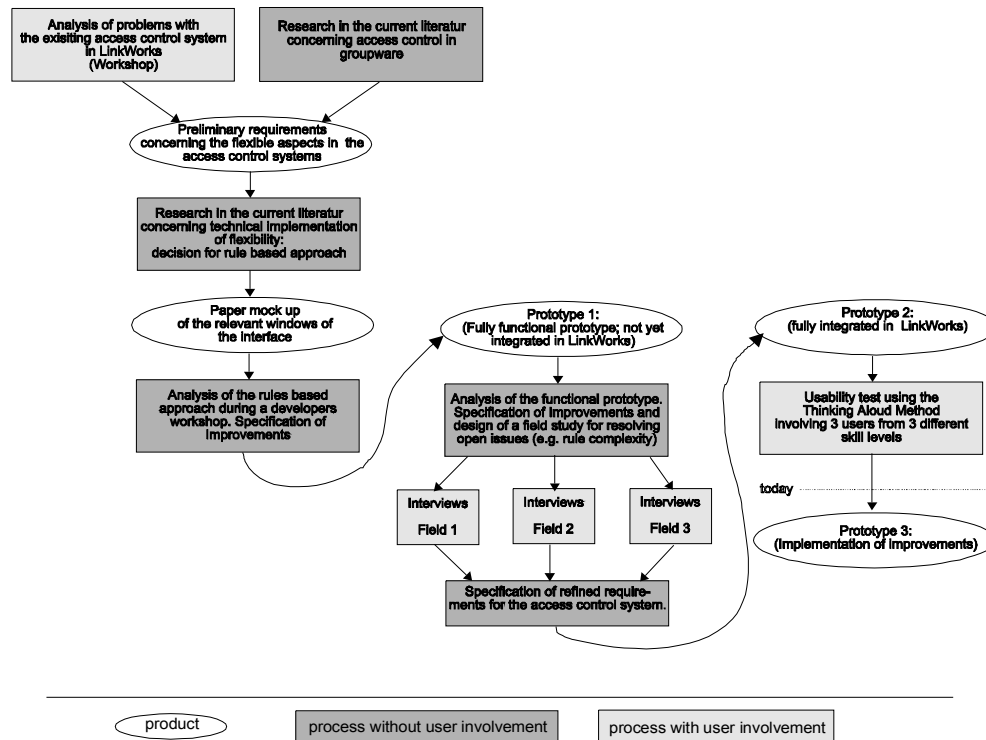


Figure 5: Design process for the new access control system

In LinkWorks (version 3.0) users can determine the access rights for an object by choosing a predefined access profile. The system provides eight default access profiles (e. g. public, private, for feedback etc.), which can be changed or extended only with a special tool, usually at the hand of the system administrator. The user advocates in the POLITEAM project reported very early on, that the users considered the existing access control system insufficient for their purposes, mainly due to the following reasons (which were also discussed and elaborated during the first user workshop):

The access profile scheme is not flexible enough, since user can only choose from a rather limited number of predefined options. If the intended access policy is not among them, end users cannot define a new one.

In LinkWorks access rights are defined in relation to formal organizational hierarchies. Therefore, it was hardly possible to implement access policies to support collaboration not following existing hierarchies.

The access rights do not recognize the desktop metaphor, e. g. it is possible to send somebody an object which this person cannot access in any way. In some cases (sending sensitive documents by mistake) this might be a desirable effect. On the other hand, users might end up with “dead” objects on their desks which they cannot remove or return to the sender.

These reasons do not concern superficial elements but stem from conceptual inconsistencies. The existing access control system does not take the diversity of different access policies into account, especially that:

- foremost the end users implement access policies, not the system administrator.
- many organizations (at least our field of application) do not primarily rely on formal hierarchies to structure and organize group processes. Workgroups may be formed orthogonal to these existing hierarchies.
- end users seem to be easily irritated by inconsistencies in the use of metaphors in the design of software, i. e. if they are presented with a virtual desktop they expect it to “behave” like a desktop. Concerning access rights, they expect their desk to be first and foremost a private place (“My desk is my castle”).

After evaluating the results of our first user workshop we decided to design a completely new access control system for POLITEAM. The object oriented architecture of LinkWorks allows for such radical tailoring by providing a high level method language. At this point we have to distinguish between the tailoring activities by the authors (tailoring of LinkWorks in order to implement a new access control system) and the tailoring activities by the end users (tailoring of the access control system in order to implement new access policies). For the purposes of this paper we refer to the former as “design” and the latter as “tailoring”, since it is not relevant here that we implemented our access control system using an existing groupware system (apart from taking into account the experiences made with the application of the existing system).

The evaluation of the problems with the existing access control system and the study of related problems mentioned in the current literature provided us with a rough understanding of the requirements of the new system. We evaluated the literature looking for a basic model for representing access policies in cooperative multi-user systems. As a result we decided to start

with a rule based approach, since it seemed to offer the necessary flexibility and power. Moreover it turned out that - during the first user workshop - our end users formulated their access policies explicitly in form of rules (permissions and denials).

One major design problem using rules was to determine the exact form of single rules, the semantics of rule evaluation and the interface for presenting and editing the rule base. We pursued a user centered approach by asking ourselves how we would expect access rules to work and be represented in the system. We designed paper mock ups of possible user interfaces and evaluated different possibilities during the first developers workshop. The result of this workshop was a natural language approach for the presentation of rules in the user interface and several interface features for effectively presenting a set of rules to the users. Concerning rule evaluation we considered a “most-specific-rule-holds”-scheme most intuitive.

The following rules are examples of typical access rules as formulated by the participants of the workshop:

- R_1 : User A is allowed to read and write documents in Folder C
- R_2 : Users of Group B are forbidden to read documents in Folder C

In the case of inconsistent rules (e.g. if User A is in Group B and tries to read a document in Folder C) the more specific rule is applied (in the example this is rule R_1 which allows access). This approach to the resolution of inconsistent rules was based solely on our (the designers) intuition about (real world) rule systems. As described later on, we refined the resolution strategy together with the users in later stages of the design process.

The next step was the design of a first functional prototype to get an idea of the problems end users might face when tailoring their access policies using rules. This prototype was implemented using Microsoft Visual Basic and allowed the editing and evaluation of a set of access rules. This prototype was again evaluated in a designer workshop, resulting in a set of minor improvements but mainly in a list of open issues which could not be resolved without the participation of end users from our field of application.

The most important issue was, upon which factors the permission or denial of access depends. This question is important for determining the necessary terms which can appear in the conditional part of a rule. If an access policy

states, for instance, that access to certain documents is to be denied on weekends, the denial depends upon the *time* of access.

The decision which factors to allow in the conditional part of a rule thus determines the range of different access policies which are supported by the system, i. e. the degree of flexibility which end users can control.

To answer these questions we decided to interview end users in different fields of application in order to determine the range of access policies which had to be supported by the new system. Eliciting the requirements from single users turned out to be rather easy, because even users with little computer experience were aware of the (common sense) need to control access to sensitive documents. All user involved in our field study could readily formulate the access policies (in their own language) needed for their work. Thus, the main challenge of this field study was to capture the full range of requirements.

We decided to include 3 different organizations in our initial survey: one of the POLiTEAM fields of application (the department of a state ministry), a private company and a semi-private organization. We selected the different organizations according the their degree of exposure to market pressures since we believed that this factor has a strong influence on the type of access policy needed. As many government departments in Germany - as a trend - are being restructured according to the ideas of customer- and service orientation (there even are several examples of “outsourcing” previously public functions like waste disposal to the private sector), we hoped to get a notion of future requirements by examining organizations with stronger exposure to market pressures. In each organization we interviewed at least one subordinate and one superior (manager). The whole classification scheme for interviewee selection is shown in figure 6:

<i>organization</i>	public	semi-private	private
<i>person</i>			
superior			
subordinate			

Figure 6: classification scheme for user selection

The interviews were conducted in the offices of the respective persons. They were semi-structured in the sense that we had prepared a set of open lead questions in order to initiate the interview and motivate the interviewee to talk about the sometimes rather touchy subject of access rights. The semi-structured questionnaire basically contained the following items:

- General questions concerning the position and responsibilities of the interviewee in the respective organisation,
- Questions concerning the (electronic and paper) documents related to the work of the interviewee (e.g.: “What documents do you work with and what do they contain?”),
- Questions concerning the collaborative aspects of the work (e.g.: “Who else needs to access these documents?”),
- Direct questions concerning the permission and denial of access to the respective documents (e.g.: “Who is allowed to read or change the documents?”).

As mentioned before, we also included questions concerning intuitive resolution strategies for inconsistent access policies (e.g.: “If one policy states that nobody is allowed to read documents on your desk and another policy states that members of a certain project group may read documents on a part of your desk, which of these two policies should be applied by the system?”).

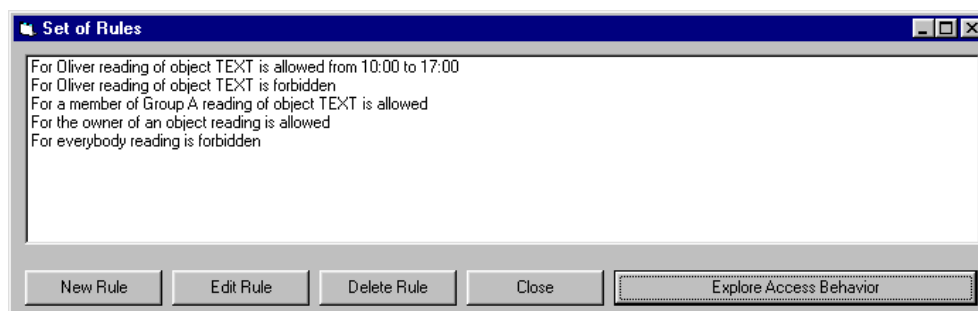


Figure 7: Screen presenting access rules to the end user

Our goal was to elicit the access policies used in connection with the documents (on paper and on existing computer media) used by the interviewees. We conducted all together 12 interviews equally distributed over the different classes in figure 6.

The main result of our survey was the set of factors - concerning the context of user and object - which were used in access policies to determine whether access is to be allowed or denied. They ranged from obvious central elements like the user or the object itself, over other anticipated factors like organizational roles to surprising aspects, e.g. the political affiliation of users, or their state of health (“Only if I am sick, my colleagues may access my desktop.”). We also noticed the important role of time-dependent access policies (e.g.: “Our sales force is only allowed to access this price list until first of March”). Since we want to concentrate here on issues of the design process of tailorable software rather than access control, we refer to our other work (Stiemerling 1996, Pfeifer & Stiemerling 1997) for a more extensive discussion of the results. We only want to mention here, that due to the rather exotic nature of some of these access factors, not all factors developed during the field study could directly be implemented in the second prototype (e.g. making access dependent on the state of health of a user, which is obviously hard to do).

Once the interviews were evaluated we began to design the second prototype. This prototype was fully integrated in the LinkWorks-environment, i.e. the access policies did have a real effect upon the documents in the system. The old access control system was neutralized. The purpose of this prototype was the end user evaluation of the rule based approach. Figure 7 shows the presentation of the rules valid for a certain object. The rules are ordered according to the interpretation algorithm with the more specific rules on top and the more general rules at the bottom.

The user has the opportunity to query the rule base (button: “explore access behavior”) in case he or she does not understand the presentation.

Figure 8 shows the screen for editing a single rule. The user enters the elements of the rule using simple, well-know control elements like drop-down boxes. A very successful feature in this screen is the *instant feedback* in natural language in the lower part of the window. After every selection in the form the rule description changes according to the users action. During evaluation this feature allowed even first time users to identify and correct mistakes.

The screenshot shows a Windows-style dialog box titled "Rule". It has a "Scope" section with five tabs: "Users", "Documents", "Relation", "Access Mode", and "Time". The "Users" tab is selected. Inside the "Users" tab, there are two fields: "Username" with a dropdown menu showing "Oliver", and "Group / Role" with an empty dropdown menu. Below these fields is a "Decision" section with a dropdown menu showing "allowed". At the bottom of the dialog is a "Rule in plaintext:" section with a text area containing the text "For Oliver reading of object TEXT is allowed". At the very bottom are "OK" and "Cancel" buttons.

Figure 8: Screen for editing a single rule

We used the Thinking Aloud Method (see Nielsen 1993) for the evaluation of our user interface. The basic idea of this method is to let users carry out a number of real world task with the prototype and ask them to “think aloud” about their interpretation of presentation and possible actions. Especially the motivation behind the actions is of interest.

The evaluation was carried out in a laboratory setting. A simple scenario involving a diary and a group of “good friends” was prepared. The users (6 users from 3 different skill-levels spanning developers, power users and novices) were given a set of tasks consisting of access policies they had to implement. A simple access policy, for example, was “The group of ‘good friends’ is allowed to read the diary.”, a more difficult one was “Oliver is not allowed to read the diary on weekends.”

The usability test revealed several problems in the design of the user interface, especially ambiguities in the use of natural language to describe rules. However, the underlying rules-concept was understood by the end users. At this point we are confident that a majority of end users in our field of application are able to successfully implement access policies using our system.

The rule based approach is flexible enough to support a wide range of access policies and we believe that we have captured a number of relevant factors determining permission or denial in real world access policies in a broad range of fields of application.

Design of a Search Tool for Groupware

The second case that shall be described here is the redesign of a search tool for the POLiTEAM project. The aspects of this case concerning the participatory and evolutionary approach of development are covered in depth in Kahler (1996).

The basic version of LinkWorks had a tool implemented that allowed the user to search for any object independent of its actual location within the system. Discussions with users revealed that this search tool was not well enough designed to be used by our application partners since the issues of privacy and unintentional manipulation of shared files were not satisfactorily dealt with, possible conflicts about snooping around on others' desks were not considered. Thus, the original design did not take the diversity of searching activities in different fields of application into account. Moreover, the user interface was overloaded with functions unneeded by our application partners. So we decided to improve the existing search tool or build a new one with the means that LinkWorks as an object oriented system provided.

Our first goal was to identify the basic functionality of a groupware search tool as well as additional features that might be needed in one organization or work setting but not in the other. To do so, in the course of the redevelopment of the search tool different techniques for requirement analysis were involved (see figure 9). We conducted 10 interviews with interview partners from four different organizations one of which was a POLiTEAM application partner, held four workshops with POLiTEAM members where aspects of searching were raised, two of which were dedicated to search tool prototypes, and we developed three prototypes of search tools which were later evaluated. Moreover, the POLiTEAM user advocates (see Mambrey et al. 1996) helped us to get a better understanding of the work of our application partners and their requirements.

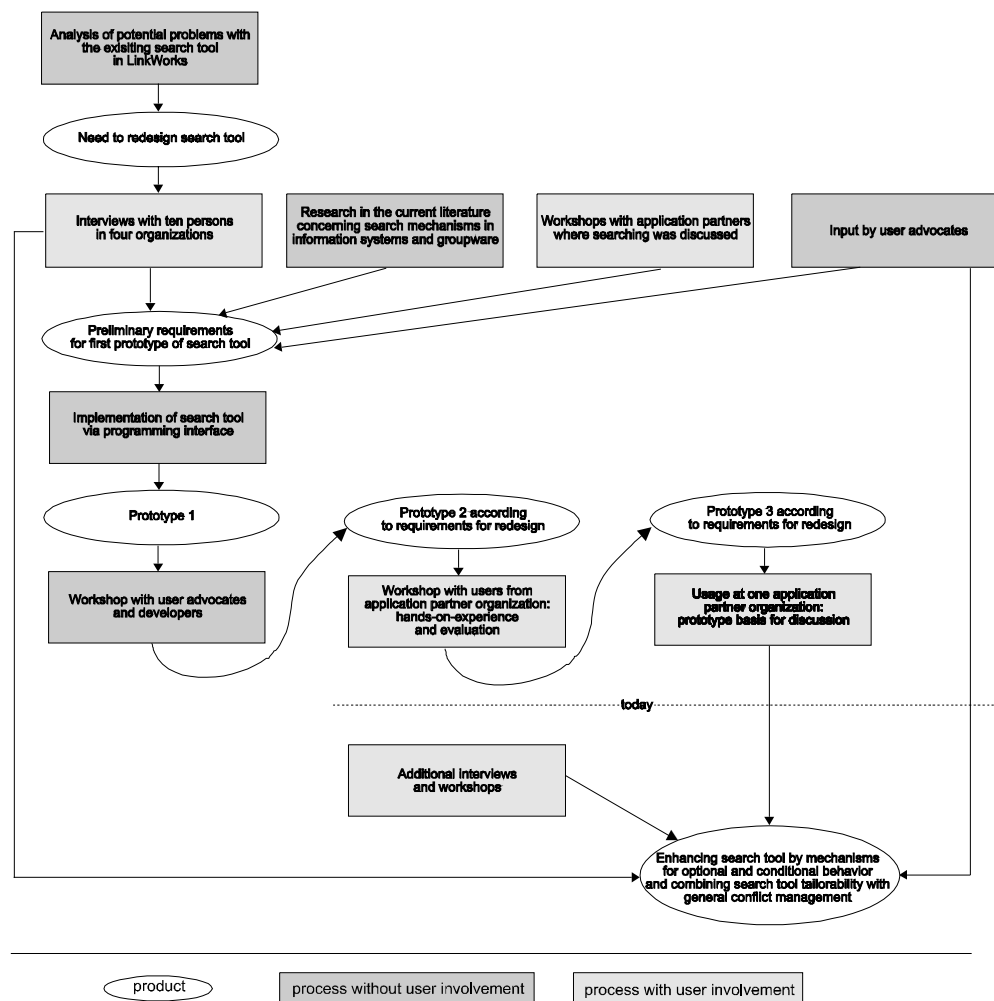


Figure 9: Redesign process for the search tool

To get a better understanding of how search in a work group is performed we started with conducting interviews about how people who cowork with each other search objects, i.e. documents, papers, or folders in an office environment. We talked to ten people, the interviews were led with one person at a time, lasted about 30-45 minutes each and were conducted along a questionnaire with 29 questions that served as a guide which left space for additional questions and talk. The questionnaire had two parts having the interviewees take the roles of both a person searching something in a work group and person „being searched on“, i.e. someone, who was asked about or for an object.

The answers of the interviewees shed a light on different aspects of searching in a work group. While the general search criteria were the same in different organizations (the file name, date, key words, and the author of a document) the ways how and where objects are stored in a particular work place differed in the different organizations. This includes organizational as well as personal storage. Several personal preferences could be found which the interviewees stated to be efficient for themselves. On the organizational level we found different structures to sort and order documents like order by date, by internal or external order numbers or by task areas and within them again by project number and date. The common search criterion to search only in text documents was later implemented in a check box of the prototype as optional behavior.

Potential conflicts came up where electronic search on others' desks was discussed. Here, the symmetric design of the questionnaire allowed for every interviewee to take the role of a „searcher“ and the role of a person „being searched on“. In the role of a person searching actively the interviewees pleaded for a nearly unlimited access for electronic search arguing that this would be helpful and necessary for cooperation and adequate for team work. When they took the role of a person affected by someone else's electronic search some of them felt uncomfortable knowing that everyone could look into their folders and considered this as an unwanted intrusion. So obviously, there is a conflict potential in performing an electronic search on another person's desk that requires a context specific solution: While in one case it might be adequate to prohibit a system-wide search at all in another case or organization it might be sensible to generally allow for a search within a work group (could be implemented as conditional behavior) or allow for it under the condition that a mail is generated that the electronic desk was searched.

Besides the interviews in this first step of the redevelopment of the search tool two workshops were held with a group of users of one application partner where searching was discussed among other topics. The workshops brought out much more of the group dynamics than the interviews were able to and underlined the relevance of different conventions e.g. regarding the naming of documents within different subunits of one organization.

In reprogramming the search tool we had to decide whether to change the original search tool to fit the new requirements or to use an external programming language for the search tool prototype and access the

LinkWorks objects by means of the programming interface. We decided for the latter alternative which lowered the performance considerably but provided for more flexibility. This was due to the fact that although LinkWorks is object oriented and has some mechanisms to change the system behavior it still had the original search tool encapsulated and did not allow to use all of the internal methods needed. A major improvement in the resulting prototype 1 was the distinction of the area where an object was found (i.e. on the searcher's own desk, on someone else's desk or in the archive of the group, see figure 10) as a first step towards conflict management.

This prototype was presented to system developers and user advocates, then changed, and the changed version (prototype 2) was shown to three users from one of our application partner organizations in a workshop with the primary goal of the evaluation of the functionality and user interface of the new search tool. These users not only suggested some minor changes to the user interface which were considered in the next prototype but also hinted at another major feature that could be subject to tailoring activities: They suggested that objects found by the search tool might not only be represented as a link to the original object on the searcher's desk but might alternatively be copied to the searcher's desk from the owner's desk. While this might be seen as a contradiction to the design ideas of LinkWorks it became clear that this was the appropriate solution for some settings.

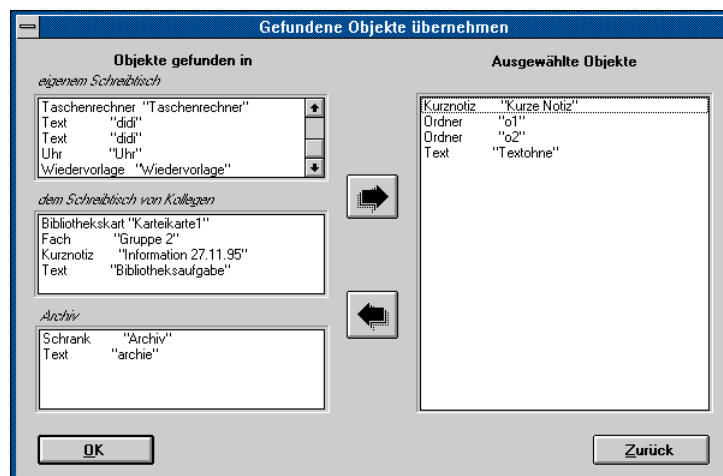


Figure 10: Output dialog of search tool showing where objects were found

Though we initially thought that this prototype 3 could become part of the POLITEAM system the feedback from the workshop and the statements of the user advocates convinced us to redesign the prototype and provide a (more) tailorable version. The current version of the search tool that suits the needs of one particular work group of our application can be considered to be the default configuration of a future system there and as a means to have the users get to experience and get used to electronic search so they can discuss refinements in the current configuration.

In a next step we will enhance the search tool by different mechanisms supporting the choice of functionality options and the construction of conditions for system behavior. To do this, the initial interviews conducted before the first prototype can be helpful to identify options for tailorability. More interviews and workshops will be held to cover the range of desired system search behavior. User advocates have stated that users in one other field of application do not like objects from their desk or certain folders to be linked or copied to certain or all others' desks but would not mind if the searcher got the information where the object is or some other attributes or would agree to a copy if they received a mail informing them about the action. This requires conditional system behavior that goes way beyond individual settings. Here, the rule-based approach taken in the design of the access control system (see first case) can be helpful to specify who may search on whose desk and if the search result is a link to or a copy of the file or just some information about it.

While the configuration of the number of search options is well-known on an individual level (e.g. in the search tool of the Apple Mac OS 7.5x) our main focus will be on the group level of tailoring where more than one person is affected by the use of system functionality. This will be subject to future intertwined work on the search tool and the conflict management to be implemented.

The redesign and reimplementing of the search tool showed that our broad approach to get requirements from different organizations by conducting interviews, doing workshops, discussing prototypes including hands-on-experience and being supported by the knowledge of user advocates helped to provide a deeper understanding of the aspects to be considered to develop search tool for groupware prepared to be enhanced by different tailorability mechanisms.

Conclusion

In this paper we have presented some thoughts and experiences on making software softer by end-user tailoring. They are motivated by the growing need for tailorability due to the diversity and dynamics of application organizations for software and particularly groupware products. We described approaches to catch these multiple requirements employing heuristic selection schemes in combination with participatory & evolutionary design techniques like interviews, workshops, user advocacy, thinking aloud, mockups and prototyping.

These methods were applied in the POLiTEAM project where we take an evolutionary and participatory approach of system development and enhancement based on Floyd's STEPS model extended by tailoring activities. We have presented the cases of the design of an access control system and the design of a groupware search tool where different methods for the tailorability requirements analysis were used.

While this work is still ongoing many questions concerning the design of groupware products for end-user tailorability remain open. Not only do we need more case experience but also a more refined taxonomy for end-user tailoring and research on software architectures supporting tailoring. The explicit integration of tailorability in existing design methodologies and modeling languages is another open question.

While the diversity of the field of application might be understood by actually researching in different organizations and subunits of one organization, the dynamics of use allowing for at least some prediction of future use is more difficult to catch empirically. Moreover, we feel that more work needs to be directed to group-related tailoring including the question of adequate default configuration, non-technical and technical conflict management and the integration of technical and organizational development. Moreover, we will have to discuss the requirements resulting from our experience for the design of a tailorable software architecture. The experiences presented here may be a starting point to link tailorability to participative system development

References

- Bentley, Richard; Dourish, Paul (1995): *Medium vs. Mechanism: Supporting Collaboration Through Customization*. In: Proceedings of ECSCW '95. Kluwer. pp. 133-148.
- Boehm, Barry W. (1976): *Software Engineering*. In: IEEE Transactions on Computers, Vol. C-25 (12). pp. 1216-1241.
- Boehm, Barry W. (1988): *A Spiral Model of Software Development and Enhancement*. In: IEEE Computer, Vol. 1988 (5). pp. 61-72.
- Carter, Kathleen; Henderson, Austin (1990): *Tailoring Culture*. In: Reports on Computer Science and Mathematics no. 107, Åbo Akademi University 1990. Proceedings of 13th Information Systems Research Seminar in Scandinavia (IRIS). pp. 103-116.
- DEC, Digital Equipment Corporation (1995): *LinkWorks Version 3.0 - User's Guide. Part number AA-Q3KYB-TE*.
- Ecklund, Earl F.; Delcambre, Louis M.L.; Freiling, Michael J. (1996): *Change Cases: Use Cases that Identify Future Requirements*. In: Proceedings of OOPSLA '96, ACM Press. pp. 342-358.
- Floyd, Christiane; Reisin, Fanny-Michaela; Schmidt, Gerhard (1989): *STEPS to Software Development with Users*. In: Ghezzi, Carlo; McDermid, John A.: ESEC'89 - 2nd European Software Engineering Conference, University of Warwick, Coventry. Heidelberg, Springer. pp. 48-64.
- Grønbæk, Kaj; Kyng, Morten; Mogensen, Preben (1995): *Cooperative Experimental System Development - cooperative techniques beyond initial design and analysis*. In: Computers in Context: Joining Forces in Design. Aarhus. pp. 20-29.
- Grudin, Jonathan (1989): *Why groupware applications fail: problems in design and evaluation*. In: Office: Technology and People, Vol. 4 (3). pp. 245-264.
- Haaks, Detlef (1992): *Anpaßbare Informationssysteme - Auf dem Weg zu aufgaben- und benutzerorientierter Systemgestaltung und Funktionalität*. Göttingen und Stuttgart, Verlag für Angewandte Psychologie.

- Henderson, Austin; Kyng, Morten (1991): *There's No Place Like Home: Continuing Design in Use*. In Greenbaum, Joan; Kyng, Morten: *Design at Work - Cooperative Design of Computer Systems*. Hillsdale, NJ, Lawrence Erlbaum Associates. pp. 219-240.
- Henderson-Sellers, Brian; Edwards, Julian M (1990): *The Object-Oriented Systems Life Cycle*. In: *Communications of the ACM*, Vol. 33 (9). pp. 143-159.
- Jacobsen, Ivar; Christerson, Magnus; Patrik, Jonsson; Övergaard, Gunnar (1992): *Object-Oriented Software Engineering: A Use Case Driven Approach*, ACM Press.
- Kahler, Helge (1995): *From Taylorism to Tailorability*. In: *Proceedings of HCI '95*, Vol. 20B. Elsevier, Amsterdam. pp. 995-1000.
- Kahler, Helge (1996): *Developing Groupware with Evolution and Participation - A Case Study*. In: *Proceedings of PDC '96*. Computer Professionals for Social Responsibility. pp. 173-182.
- Kjær, Arne; Madsen, Kim Halskov (1994): *Participatory Analysis of Flexibility: Some Experiences*. In: *Proceedings of Participatory Design Conference '94*. pp. 21-31.
- Klöckner, Konrad; Mambrey, Peter; Sohlenkamp, Markus; Prinz, Wolfgang; Fuchs, Ludwin; Kolvenbach, Sabine; Pankoke-Babatz, Uta; Syri, Anja (1995): *POLITeam: Bridging the Gap between Bonn and Berlin for and with the Users*. In: *Proceedings of ECSCW '95*. pp. 17-31.
- MacLean, Allan; Carter, Kathleen; Lövsstrand, Lennart; Moran, Thomas (1990): *User-Tailorable Systems: Pressing the Issues with Buttons*. In: *Proceedings of CHI 90*. pp. 175-182.
- Malone, Thomas W.; Lai, Kum-Yew; Fry, Christopher (1992): *Experiments with Oval: A Radically Tailorable Tool for Cooperative Work*. In: *Proceedings of CSCW '92*. pp. 289-297.
- Mambrey, Peter; Mark, Gloria; Pankoke-Babatz, Uta (1996): *Integrating user advocacy into participatory design: The designers' perspective*. In: *Proceedings of PDC '96*. Computer Professionals for Social Responsibility. pp. 251-259.
- Mørch, Anders I. (1997): *Method and Tools for Tailoring of Object-oriented Applications: An Evolving Artifacts Approach*. University of Oslo, Department of Informatics. Dr. Scient. Thesis Research Report 241.

- Nardi, Bonnie A.; Miller, James R. (1991): *Twinkling lights and nested loops: distributed problem solving and spreadsheet development*. In: Int. J. Man-Machine Studies, Vol. 34. pp. 161-184.
- Nielsen, Jakob (1993): *Usability Engineering*. Boston, Academic Press.
- Oberquelle, Horst (1994): *Situationsbedingte und benutzerorientierte Anpaßbarkeit von Groupware*. In Hartmann, Anja; Herrmann, Thomas; Rohde, Markus; Wulf, Volker: *Menschengerechte Groupware - Software-ergonomische Gestaltung und partizipative Umsetzung*. Stuttgart, Teubner. pp. 31-50.
- Olsen, Dan R.; Foley, James D.; Hudson, Scott E.; Miller, James; Myers, Brad (1993): *Research Directions for User Interface Software Tools*. In: Behavior and Information Technology, Vol. 12 (2). pp. 80-97.
- Paetau, Michael (1994): *Configurative Technology: Adaption to Social Systems Dynamism*. In Oppermann, Reinhard: *Adaptive User Support - Ergonomic Design of Manually and Automatically Adaptable Software*. Hillsdale, N.J., Lawrence Erlbaum. pp. 194-234.
- Pfeifer, Andreas; Stiernerling, Oliver (1997): *Konfiguration des Informationsdienstes in Groupware*. In: Proceedings of Wirtschaftsinformatik '97. pp. 263-278.
- Stiernerling, Oliver (1996): *Anpaßbarkeit von Groupware - ein regelbasierter Ansatz*. Department of Computer Science III, University of Bonn. Diploma Thesis.
- Trigg, Randall H. (1992): *Participatory Design meets the MOP. Accountability in the design of tailorable computer systems*. In: Proceedings of 15th Information Systems Research Seminar in Scandinavia (IRIS). pp. 643-656.
- Trigg, Randall H.; Moran, Thomas; Halasz, Frank (1987): *Adaptability and Tailorability in NoteCards*. In: Proceedings of Human-Computer Interaction - Interact'87. pp. 723-728.
- Wulf, Volker; Rohde, Markus (1995): *Towards an Integrated Organization and Technology Development*. In: Proceedings of Symposium on Designing Interactive Systems. Processes, Practices, Methods & Techniques. ACM. pp. 55-65.

Third Paper

More Than WORDs - Collaborative Tailoring of a Word Processor

Abstract

Tailorability (or adaptability) of software becomes more important with the increasing use of off-the-shelf-software. On the other hand, computers support the work of many groups which in turn have to tailor a commonly used software to support individual as well as group needs. This includes not only groupware, i. e., software that directly supports collaborative work, but also single user software. Research has shown that often adaptations to single user software are distributed among colleagues, thus leading to a systematization in a group's adaptations. Based on this observation an empirical field-study on the collaborative tailoring habits of users of a particular word processor was carried out. Based on these and literature research an add-on to this word processor was developed which provides a public and a private repository for adaptations as well as a mailing function for users to exchange adaptations. Some notification and annotation mechanisms are also provided. Results of two forms of evaluation indicate that users of different levels of qualification are able to handle the tool and consider it a relevant alternative to existing mailing mechanisms.

Introduction

Generic single user applications for obvious reasons do not provide support to share adaptations (i. e. the results of a tailoring activity, tailoring artifacts) among their users. However, they are often tailored collaboratively. Complex tailoring is carried out individually or even jointly and distributed among colleagues. Particularly with the increasing number of group or organization wide computer networks such a form of collaborative tailoring seems promising in two ways: Firstly, double work can be avoided if

adaptations that are helpful for several persons are made once only and then distributed. Secondly, sharing adaptations among groups of users can lead to a systematization of adaptations avoiding a confusing abundance of individual solutions. Therefore, I set out to develop design suggestions for a tool to help people to collaboratively tailor software.

In order to do so resources from different fields were gathered: There has been work conducted dealing with tailoring of software and particularly tailoring of word processors. Moreover, in CSCW (Computer Supported Cooperative Work) much effort has been spent to understand how people collaborate and several authors have provided empirical information and theoretical background on collaboration and particularly on tailoring a commonly used software. And finally, some work in the field of CSCW and Information Systems has been devoted to the evolving use of information and communication technology in organizations. While this paper does not provide a longitudinal study of such an evolving use, I used the idea as a starting point by attempting to understand how the use of a particular word processor had evolved in several different organizations.

Related Work

Tailoring Software

Tailoring software is not a new phenomenon. More than 20 years ago the EMACS editor provided mechanisms for extension by the user while it was running (Stallman 1981). Since then, several authors have dealt with the issue of tailorability of software with a background of Human-Computer Interaction (HCI). According to Mørch (1998) tailoring is the activity of modifying a computer application within the context of its use and can be considered to be further development of an application during use to adapt it to needs that were not accounted for in the original design. Henderson & Kyng (1991) also consider tailoring to be an activity that continues design in use. They argue that there is a necessity to be able to change a system after its initial design due to the change of use situations, the complexity of the world that makes anticipation difficult, and different situations that one software might be used in. Haaks (1991) distinguishes different dimensions of tailoring including initiator and actor, object, aim, time, and scope of validity. All of these authors stress that the discussion about tailoring should not only be lead in terms of technical measures, but that tailoring software is

an activity that is deeply rooted in personal habits and preferences as well as socio-organizational circumstances and dynamism. In his plead for situative tailoring and local activities Paetau (1991) explicitly distinguishes between different forms of cooperation and introduces the concept of *cooperative configuration* where tailoring of a technical system is to be considered as a basically cooperative process.

Tailoring software can be distinguished from use and development although it bears similarities with both. On one hand it is a way to continue design in use to account for unanticipated needs, on the other hand it extends use by providing means to make it effective and efficient. Henderson & Kyng (1991) argue with the relative stability of an application in claiming that people tailor when they change stable aspects of an artifact. However, they also admit that the distinction may be difficult: Changing the font of a document can be considered to be use or tailoring. They also introduce the notions of subject matter vs. tool of work and claim that changing the subject matter is use while changing the tool is tailoring. Again, the distinction is not always clear, since one person's subject matter is another person's tool: For a person using an application programmed in C++, this application is a tool, whereas for its programmer it is the subject matter, and the C++ compiler is the tool (and for the compiler builders it is the subject matter). So if someone's main activity is using a text editor to produce text and she writes some macros with a built-in macro-editor to make text writing easier for her this writing macros is considered tailoring. If however, her main activity is writing macros for the text editor for the sake of the intellectual challenge or to provide a service to someone else this is considered use of the text editor and its built-in macro-editor rather than tailoring. That way sometimes tailoring may turn into use when a person who does a good job writing macros for a text editor starts to do this for a whole group of users so that finally the time and effort to write macros outweighs the time and effort to produce text related to the original task of that person in the group. Since this paper's focus is on end users who perform their primary work task and do some tailoring once in a while, the more advanced endeavors for the distributed development of Linux or the creation and maintenance of an EMACS Lisp library or an EMACS widget library shall be considered use of Linux and EMACS and the advanced tools related to them rather than tailoring: for a programmer programming is not a new dimension.

Some work in the HCI area has particularly focused on tailoring word processors. Page et al. (1996) investigated the tailoring habits of users of word processors by means of a quantitative study surveying word processor usage and tailoring of 101 people over 28 days. They recommend to expect users to tailor the software and require that “tailoring features become an integral part of the system and its user interface” (p. 345). Cypher (1993) reports that macro recording in a word processor can effectively automate many repetitive user activities. Both contributions, however, do not focus on the collaborative aspects of these activities.

Collaborative Aspects

In the CSCW literature tailorability has been identified as a key requirement for groupware systems (see, e.g., Bentley & Dourish 1995, Oberquelle 1994, Stiemerling et al. 1997). The special demands of collaborative work make it a critical issue in the design of groupware applications. Complexity, dynamism, as well as inter- and intra-individual differences constitute the need for system designs, which can evolve over time, exhibit different behavior in different usage situations, and accommodate individual or group needs and preferences.

On one hand several suggestions have been made for groupware architectures and technical approaches to tailorability. On the other hand collaborative aspects of tailoring have been observed and discussed in different fields (see, e.g., JCSCW 2000 for both). Some research has also been concerned with collaborative aspects of single-user software.

The contributions up to date are primarily of observing nature. Mackay (1990), for instance, describes how users of different qualification levels exchange customization files. While writing such a file from scratch demands a high level of qualification, simply using a file copied from a colleague is quite easy. She describes different “patterns of sharing” such files in real world fields of application. In Mackay (1991) triggers and barriers for customizing based on data from 51 participants working in a UNIX software environment are described. Nardi (1993) presents the result of two field studies concerning collaborative tailoring (*end user programming* in her terminology) among spreadsheet and CAD users. She views collaborative tailoring as a natural consequence of the division of labor and stresses that this aspect of tailoring has to be taken into account in the design of software systems.

Other work investigates collaborative tailoring in an organizational setting. Carter & Henderson (1990), for instance, postulate the necessity for a "tailoring culture" within an organization. They argue that tailoring not only poses technical problems, but since tailoring changes the way individuals and groups work, a culture has to be created in which technical and organizational change is something everybody can participate in and contribute to. Trigg & Bødker (1994) found an emerging systematization of collaborative tailoring efforts in a government agency. In their study they were looking at the tailoring of word processors.

Few contributions do not only observe and analyze but also take collaborative tailoring into account in the implementation of software systems. The first of these is presented by MacLean et al. (1990). The authors describe the "Buttons" system, the main tailoring entity of which are button-like objects. These objects are designed to be sent around the office by email. Thus, more experienced users who tailor, e.g., the lisp-code behind a button, can share these adaptations with their colleagues. However, while the "Buttons" system was actually used even in the non-academic parts of the research institute, it was restricted to the Xerox InterLISP environment and therefore was not exposed to users in other types of organizations. The Tviews approach (Wasserschaff & Bentley 1997) allows users to define different views on a commonly used object. Those tailorable views serve as means to show selected attributes of an object and their changes, e.g., indicate, that a shared document was changed by another person. The tailored views can be stored, retrieved and manipulated like other files via a shared workspace. However, the approach and its implementation are presented without evaluation.

Evolving Use

One of the arguments for tailorability of software is the impossibility to anticipate the future use of the software. This is due to changing task requirements, changing individual preferences, and changing group and organizational structures but also to the fact that individual and particularly group use of software is subject to evolution per se. Individual users and groups become more experienced with the software, they might find ways to use it that had not been foreseen by the software developers and they find out about the interrelation of the software they use with their task and organizational setting and how they can and do change each other. Taking

this into account, there is a growing debate about *evolving use of software* particularly in the field of information systems (IS) and CSCW (see, e.g. JCSCW 2001). The contributions stress the situatedness of all work and aim to understand the forces driving this evolution. Orlikowski (1996) found out that in an organization using Lotus Notes both planned and emergent changes in use appeared. An organizational solution for the distribution of labor between people working with Notes in the front- and backoffice of customer care could only be found after a while of use of the system when people had understood what they could do with the system and how it had changed and possibly could change the work and distribution of labor. Wulf (1999) describes how in a section of a German federal ministry the common work on text documents like manuscripts of speeches of the federal minister, answers to inquiries from the parliament or answers of letters sent by citizens changed when computers for the section members and a groupware system were introduced. Before this the texts were handwritten by members of the section and then typed by a member of a typing pool, checked for mistakes or changes to be made by the person who had originally written it and then (partially) retyped by someone from the typing pool. After the introduction of desktop computers to the section members they started to type shorter documents themselves which after a while lead to a restructuring of the division of labor. While this was considered to be more efficient than the previous state it also meant an increase of the workload of the section members and a decrease of workload of the typing pool that finally might lead to a cutback of jobs there. While it can certainly not be foreseen how software use will evolve in a particular organization I agree with Stiemerling et al. (1997) that it is necessary to look at different possible use situations in order to get a broad although incomplete perspective. Since I did not feel that the existing literature provided enough material on actual use and collaborative tailoring of a word processor, interviews with users seemed a good way to broaden the perspective (see section *Empirical Pre-Study*).

A Next Step

So far, the analytical achievements of understanding collaborative tailoring within different settings had not yet lead to an implementation of mechanisms to support collaborative tailoring of a generic widespread single-user software. In the work presented this is provided and the question is investigated how collaborative tailoring of real world applications can be

supported by technical mechanisms. I have taken Microsoft Word 97 as an example of a widely and extensively used product. As a first step, in order to learn more about how groups of users actually tailor collaboratively, a field study in four different fields was undertaken. The result of the study are a number of different collaborative tailoring use situations focusing on the exchange of document templates and toolbars. Based on an analysis of these use situations requirements for the design of a tool were developed and implemented as Microsoft Word 97 add-in which provides collaborative tailoring functionality. The implementation of the prototype is described in section *Implementation*. The use situations drawn from the field study also serve as a basis of the evaluation of the prototype which is described in section *Evaluation*. The paper concludes with directions for future work.

Note, that this paper is not about groupware but about groups of users using the same software and thus being able to employ the fact that this software is tailorable to collaborate. While this does not exclude groupware it encompasses a much broader range of (single user) software. Oberquelle (1994) proposes a classification of collaborative tailoring (in his work only related to groupware) where he distinguishes between actors, who can be individuals or a group and persons affected by a tailoring activity, who can again be individuals or a group (see figure 1). Different aspects and intensities of collaborative tailoring of a single user software can fit in all of the resulting four categories. Examples are given for a word processor:

- Individuals can tailor for themselves and are the only ones affected by the tailoring activity – e. g. individual keyboard shortcuts or the window layout of an individual email client (quadrant I).
- Individuals can tailor for a whole group who then agree or are obliged to use the adaptations – e. g. a system administrator or expert user provides a letterhead to be used by the group (quadrant II).
- A group can tailor synchronously or asynchronously and its members agree or are obliged to use the adaptations – e. g. several persons work on a letterhead to be used by the group (quadrant III).
- A group can tailor synchronously or asynchronously for its members to use and change the adaptation – e. g. several persons work on collection of macros that individuals can use and change (quadrant IV).

		Actors	
		Individuals	Group
Persons Affected	Individuals	I Individualization	IV Individualization supported by group
	Group	II Tailoring effective for group	III Group tailoring

Figure 1: Classification of collaborative tailoring following Oberquelle (1994)

This contribution provides examples for different forms of collaborative tailoring and introduces a tool to support these for a word processor.

Empirical Pre-Study

To learn about users' habits and to inspire the design, a qualitative field study with users of Microsoft Word was carried out. 12 semi-structured interviews with users from 4 different fields were conducted (public administration, private company, research institute and home users).

The interviews started with general questions about the interviewee's qualification, their general task and the way they apply the word processor. In the following they were asked which tailoring functions were in use, which barriers hindered the usage of existing functions to tailor, whether and how collaborative tailoring did take place, and whether organization wide standards concerning the tailoring activities are existing. In the end of the interviews ideas concerning the design of support for tailoring activities and of improved user interfaces to ease tailoring were discussed. To be able to refer to the software, the interviews were performed next to the interviewee's computer.

The interviews took between 20 and 120 minutes with an average of about 45 minutes. They were audiotaped, transcribed and later analyzed. According to their self-estimation two interviewees had little to medium, two interviewees had medium, three interviewees had medium to good, three interviewees had good and one interviewee had very good knowledge

about the word processor. Five of the interviewees were providing system support to other users in their organizations.

Together with the literature review these interviews are the basis for the requirements. The interviews are enriched by empirical studies concerning the usage and sharing of a tailorable search tool for groupware. A prototype of this search tool was presented, used and discussed in a workshop with users of the representative body of a German state government where some of the interviews about Microsoft Word had taken place. The workshop about the search tool and interviews about it revealed interesting aspects of sharing adaptations in this organization.

Depending on their field of application the interviewees reported about differences in the extent and the way tailoring is seen as a collaborative activity. To give an impression of this variety and to motivate the design, I will present the main statements of the interviewees concerning collaborative tailoring.

Use Situation I: Central Repository for Standardized Forms

One group of persons interviewed were two system administrators and two researchers from a German national research institution. The system administrators were responsible for supporting the Unix and the PC environment in one of the subunits of the research institution. The researchers were employees of the same subunit working in two different research groups.

The interviewees reported about little collaborative tailoring activities. Since the members of this subunit employ a rather heterogeneous spectrum of word processors and software versions, since their tasks are rather individualized, and since most of them are rather experienced with the system, they participate in little collaborative tailoring. Nevertheless the organization uses an intranet to provide certain document templates in a standardized manner. The members of the organization find document templates of administrative purpose on one of these intranet servers (e.g., ordering and billing forms). These templates are created and updated by a central organizational unit, which has been built up recently. All the other users can just copy these templates. Ideas for new forms have to be proposed to that unit. This is an example of “tailoring effective for the group” in figure 1.

This quite centralized view of sharing adaptations is similar to the situation found in the search tool workshop. However, in the search tool case it would have been possible for all participants to tailor and share but they argued that for reasons of an adequate division of labor it would be sensible for the colleague who provides local computer support, an administrative clerk, to tailor the search tool and provide different versions among which the others would then only choose without tailoring themselves. This “local expert” later argued in an interview that he would like to have his own private corner where he could work on different search tools and store incomplete versions without the others being able to access them.

Use Situation II: Collaborative Tailoring and Organization-Wide Distribution

Four of the interviewees were working for the representative body of a northern German state in the federal capital. The organization had been equipped with generally available desk-top computers about three years ago. Two of the interviewees were heading sections responsible to represent the interests of their state within the process of federal legislation. The other two were working in the administration of the body. One of them provides system support to the other users.

All of them reported about a rather intense exchange of adaptations. One of the employees from the administration site reported how she created a document template together with a colleague. Both of them carried out parts of the whole job. Then she put her part of the template on a disk and carried it to her colleague who pasted the parts together. This could be considered to be “individualization supported by group” in figure 1 or even “group tailoring” if after a while of usage everyone agrees to make this template their standard.

In the representative body there is not a formal procedure on how to decide on commonly used document templates. One of the employees reported that it is often a difficult task to find a consensus. At the times the interviews were conducted, templates were printed out and handed over from employee to employee. Each of them could annotate the printout. The interviewee being responsible for the creation of document templates was often overwhelmed by the inconsistent feedback and found it difficult to decide on the final layout. In cases she could not satisfy all of the requirements, she recommended her colleagues to create individualized versions of the

template on their private desk. Thus, the process to create document templates was rather unstructured.

To make document templates publicly available, the representative body used the groupware system whose functionality offered shared workspaces to exchange documents. To publish newly created document templates within the whole organization, a specific workspace was used. Within this workspace simple users just had the right to copy documents. Because several of the employees suffered from lacking computer skills, the right to change these templates or to add in new templates was reserved to the system administrator. The templates were seen rather as a collective resource than as a means to enforce organizational standards.

Use Situation III: Shared Document Templates and Notification of Users

An experienced user working in the marketing division of a car-manufacturer described how he had implemented department-wide standards for presenting certain data by means of tables. Before, everybody in "his" department had used his own mode to create these tables. He started to standardize the layout of these tables by creating a first template containing some macros. He then discussed it with his colleagues. Having found an agreement with them, he asked his boss for a final approval. In the end he put the templates on the LAN giving most of the workers of his department read and write permission. One of the users of whom he thought that he would endanger the template due to lacking skills was just granted read permission. Read permission was given to another user from a neighboring department who was interested in that template for his purposes. When everything was set up, the interviewee informed his colleagues verbally about the location of the shared template on the LAN. This could be considered to be another example of "tailoring effective for the group" in figure 1. One could also argue that discussion of the templates among colleagues makes this "group tailoring" according to figure 1.

Obviously, when adaptations are distributed via a shared directory, it is crucial to inform the other users. Along the same lines, a system administrator reported that he put a notice on the department's black board to inform his colleagues about newly created document templates.

Use Situation IV: Experience Transfer Among Insulated Home Users

The interviewees working at home were two law students. They used their word processor to work out law cases, which they had to deliver for getting certain credit points. Each student has to write these papers almost every semester by himself. Such a paper contains about 30 typed pages. Moreover, both students used the word processor for typing letters of different kinds.

The students reported about few collaborative tailoring activities. One of them describes these rare occasions as follows. Occasionally when he meets other students applying the same word processor he sees an unknown tailoring feature – for instance a new toolbar. In such a case he asks how the feature has been constructed. After receiving a demonstration he goes home and tries to repeat on his own system what he has seen before. Considering the classification of figure 1 this can be regarded as enhanced “individualization” where one person’s solution is used in parts by one other person or as a first step towards “tailoring effective for group”.

Discussion

Looking at the different use situations quite a wide variety of collaborative forms to tailor word processors covering the classification of figure 1 can be found.

While use situation IV just deals with experience transfer, use situations I to III are based on an exchange of adaptations. In these use situations, this common use of adaptations is either technically non-supported (exchange of floppy disks) or supported by tools, which are realized apart from the word processor (intranet, LAN directory, groupware application). Both of these solutions seem to be problematic because they require the users to leave the application to acquire the adaptations. Therefore, it seems worthwhile considering to integrate support for collaborative tailoring into the word processor’s functionality.

To design such an integrated support, the following considerations seem to be of special importance. Depending on the state of a tailoring activity there are different groups of users involved in carrying them out (e.g., use situation II). The extent to which adaptations are reasonably shared obviously corresponds to the tasks which are supported by them. Such a task

can be specific to an individual (e.g., use situation IV), a group or a department (e.g., use situations II and III) or even a whole organization (use situation I).

Thus, support for collaborative tailoring should allow differentiating among various groups of users when sharing adaptations. Sharing of adaptations can require different mechanisms. There are obviously situations where mail support seems to be appropriate to exchange adaptations. Use situation II presents such a case where users are jointly building a document template. Moreover, in cases an experienced user builds an adaptation especially required by a user for the task at hand, a mail tool seems to be the appropriate technical support for distribution. On the other hand, in case adaptations are not required instantly by a specific user, a publicly accessible store allows to select among these adaptations at the moment required by the task at hand (e.g., use situations I to III).

Finally there is a need to make users aware of the fact that somebody else has produced an adaptation with relevance to them. Right now users get notified verbally or by a notice on the black board (e.g., use situation III). An integrated tool to support sharing of adaptations could provide additional awareness within the system.

In the end of the interviews, the design of support for collaborative tailoring activities was discussed with the interviewees. Two main design issues emerged during the discussion. First, several interviewees suggested hiding the underlying directory structure of the tool from the users. An experienced user put it this way "The people get crazy when they have to look for something on drive M:\.. (...) You find out that every user just wants to store. If he needs something he just wants to load. He does not care at all where it is from." Such a hidden structure obviously eases the handling of such a tool. Moreover, some of the interviewees asked to store the shared adaptations in an anonymous way. One user argued this way: "Information about the creator of the tailoring data are a mess. (...) He should be anonymous in the public network because otherwise someone says 'What have you done there? That is ridiculous!'" While this argument tries to protect the creator against criticism from other users it nevertheless does not consider that creators of successful adaptation may get positive feedback. The mode to handle this issue seems to depend on the tailoring culture of an organization – especially whether it is possible to reach organizational appreciation by providing useful artifacts.

The use situations also show that the categories of that classification need some refinement considering for example the question what “the group” should be (more than one person of the group or necessarily all members) and the issue of different intensities of collaboration (using or changing someone else’s adaption vs. equally distributed work on an adaptation).

Design requirements to support collaborative tailoring

Evaluating the use situations and summing up the results of the final discussion with the interviewees, the following main requirements for the tool emerged. It turned out that this empirical evidence is in line with theoretical and empirical work described in the literature about tailorability:

- tight integration in the word processor application (see Henderson & Kyng 1991, p. 233: “most interesting are mechanisms that are itself part of the system”);
- mechanisms for sharing, sending and receiving adaptations
 - a public store to provide a location to exchange adaptations (see Bentley & Dourish 1995, p. 145: “it is possible to add attachments to the shared workspace for others to retrieve and use”);
 - mailing mechanisms for users to be able to send adaptations directly to other single users and groups of users (see MacLean et al. 1990, p. 178: users “can send a button to someone else by email”);
 - a private store for adaptations that may be copies of files from the public store or files received from others via the mailing mechanism;
- an awareness service which notifies users about modifications on adaptations (see Henderson & Kyng 1991, p. 233: “news must be published that change is available”).

The use situations indicate that the document templates are probably the most widely used adaptations in Microsoft Word 97. According to the interviewees, toolbars are an other function of word processors, whose tailoring is perceived being rather beneficial. Therefore, I decided to focus on this part of the functionality by extending Microsoft Word 97 to support the sharing of adaptations.

Implementation

In this section the options that Microsoft Word 97 in the default version provides to create and exchange adaptations are explained. Then the implementation of the first prototype based on the requirements stemming from the analysis are described. First I focus on the basic architecture of the system. Then I present the different sharing strategies offered by the tool. Finally, the questions of privacy, finding or identifying adaptations, and the implementation of a notification service are discussed.

Default Options for Adaptations in Microsoft Word 97

Microsoft Word 97 provides several options for users to tailor it to their needs. The menu item *Tools/Options* allows for the activation or deactivation of numerous check boxes thus "choosing between alternative anticipated behaviors" in the terminology of Henderson & Kyng (1991). These parameters concern, e.g., the options for saving, printing or spell checking. Their settings belong to one executable program and are saved to be accessible only to the system (e.g., in the registry of Microsoft Windows NT). They cannot be extracted or given to others by an averagely experienced user. The menu item *Tools/Customize* allows for the modification and creation of menu items and toolbars and the assignment of key shortcuts. Moreover, it is possible to tailor on a higher level and "construct new behavior from existing pieces" (Henderson & Kyng 1991), e.g., by recording keystrokes and other actions to create a macro. Such a macro can then be manually edited in Microsoft's Visual Basic for Applications or created completely from scratch. Since the macro consists of Basic code it is basically possible to extract a macro and send the ASCII text to someone who can then incorporate it for their own work with Microsoft Word 97. Macros, toolbars, styles and AutoText can also be saved as part of Microsoft Word 97's document templates (.dot files). These templates are very similar to Microsoft Word 97 document (.doc) files and can also include ordinary text or forms to fill out. Like documents the document templates can be saved as separate files and can therefore be exchanged, e.g., by email or floppy disk. Moreover, it is possible to copy macros, toolbars, styles, and AutoText between document templates (menu item *Tools/Templates and Add-ins*). That way, document templates and the included tailoring information that are located in a shared directory can be used by all persons having access to that directory.

This functionality for adaptations coming with Microsoft Word 97 is obviously intended for the use of single persons but not meant to support groups in joint tailoring. While it is possible to extract, share, and reuse some of the tailored functionality there are no mechanisms for explicitly working together on adaptations, sharing or sending them, commenting them or notifying others about useful adaptations that one might consider helpful for them. In the next section the architecture chosen to enhance Microsoft Word 97's functionality accordingly is described.

Basic Architecture

The prototype was developed in VBA (Visual Basic for Applications), the Microsoft application macro language which allows direct access to the object model of the application and offers language elements and components for the design of graphical user interfaces.

The exchange of toolbars and document templates is done transparently (in the technical sense) for the user via the distributed file system of the operating system. The application logic resides completely on the client side. Figure 2 shows the basic architecture of the system.

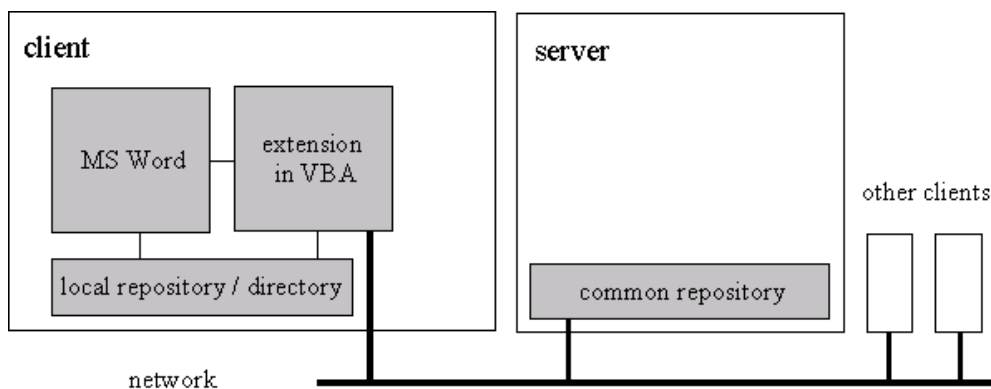


Figure 2: Architecture of the prototype

The extensions are integrated in the Microsoft Word 97 menu bar in order to make it easy for users to access the tailoring system. The basic functionality comprises loading and saving document templates and toolbars. It is also possible to combine a document template and several toolbars in a package, intended for a certain word processing task, e.g., design of a web page or

writing of a mathematical paper. The functionality is provided by a Microsoft Word 97 add-in (labeled "extension in VBA" in figure 2).

Sharing document templates and toolbars

The prototype offers both a sending and an access mode for sharing adaptations. In order to support centrally administrated environments, adaptations can be sent to groups of users. This operation might be used by administrators equipping all Microsoft Word 97 installations with a new corporate letterhead. The operation can also be used by users to mail, e. g., a certain template to a specific colleague.

It is also possible to simply store the adaptations in a shared workspace ("common repository" in figure 2). If another user is searching for a certain adaptation she can access the required templates or toolbars in the common repository.

The existing functionality of Microsoft Word 97 concerning adaptations and the extensions provided by the tool can be found under a new item in the main menu named *Adaptations*. While there are still good arguments against such a central collection of tailoring options I followed Oppermann (1991) who argues in his comparison of situated and anticipative tailoring that a dedicated menu item increases the chance that users remember the possibility of tailoring options and how to find them. The prototype related entries in this menu allow for saving toolbars only or for saving toolbars together with document templates, starting the notification and starting the adaptation browser.

Figure 3 shows the adaptation browser which offers send and access functionality to the user. It can simply be accessed via the "Adaptations" menu of the word processor.

On the left side one can see the content of the shared workspace, while the private, local repository is shown in the middle. The two lists on the right side show the other users in the system and user groups. In the screen the user can select adaptations and move them between local and shared workspace or send them - as describe above – to single users and groups. For repetitive mailing, groups of users can be defined and maintained (lower right of figure 3).

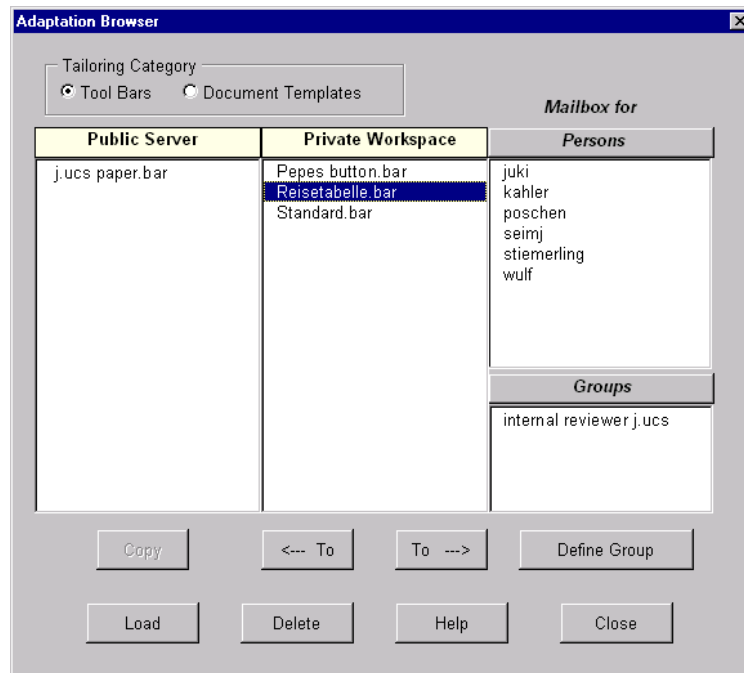


Figure 3: The browser to share adaptations (screenshot translated from German)

Identifying adaptations in common workspaces

Using the access mode of sharing requires the identification of relevant adaptations in the eventually rather large common repository. To this end the prototype offers three features.

First, it is possible to annotate an adaptation with a *textual description of its rationale*, e.g., describing the circumstances or tasks for which it might be useful. The description is displayed when browsing both repositories. The possibility to annotate the adaptations and particularly the need for annotations that were commonly understandable had been stressed in the search tool workshop in the state representative. Moreover, since elaborate adaptations are usually not self-explanatory and often closer to programming than to just choosing from some alternatives it is sensible to learn from the experiences in groups of programmers where commenting your code is mandatory. Furthermore, it is possible to identify the author and the date of an adaptation with the tool.

Second, concerning button bars, a preview mode was implemented, allowing the quick instantiation (and removal) of toolbars on the screen. This feature is supposed to appeal to more visually inclined users. The users can immediately explore the alternatives.

Third, when searching in the private or public store the users can select which categories of the adaptations should be displayed by the browser. Currently it is only distinguished between toolbars and document templates, but I believe that a more differentiated categorization might be useful, especially if the tool is supposed to scale for larger organizations. It might be even be necessary to go as far as providing a tool for logical search (based on certain attributes of the different adaptations).

Notifying the user of the arrival of adaptations

The send mode of sharing makes it necessary to inform the users when new adaptations have been mailed to them. Otherwise the available document templates and toolbars might not conform to their expectations which leads to confusion. Therefore a simple notification service was implemented which informs the user via a message window at start up time of the word processor and at the time the user activates a tailoring function in the menu. This window presents the adaptations and asks the user either to store it in his private repository or to delete it instantly. Currently, the user is notified when she starts Microsoft Word 97 and when she enters the *Adaptations* menu.

Privacy aspects

In this prototype I have strictly distinguished between a private and a common repository for adaptations. In organizations with intense internal competition, certain successful (e.g., in the sense of time-saving) adaptations are regarded as precious assets by their inventors and thus are considered worth a certain degree of protection. Therefore the private repository is located on the local machine and cannot be read by remote adaptation browsers. The common repository is right now based on the idea of equal access rights for all its users. Any user can make his adaptations available by storing them in the common repository. This repository is accessible by any user.

Evaluation

In this section the evaluation of the prototype is described. It consisted of two parts, a usability test and a quantitative evaluation.

Usability Test

The goal of the usability test was twofold. On one hand I wanted to find out if and how well the users taking part in the usability test understood the concept of sharing adaptations and the way it was implemented in the prototype. On the other hand I expected some hints for the improvement of the prototype. The usability test took part in two sessions with a team of two participants at each session. Three of the participants had been interviewees in the empirical pre-study, one of whom was the experienced user of use situation III and another one an administrator from use situation II. The sessions took place at our research site. Each of the sessions lasted about two hours. Besides the participants two observers took part in the usability test session. Each session was recorded on an audio tape to allow for clarification of what was said after the test. The sessions consisted of three parts. In part one the participants were explained what they could do with the sharing tool and were given a sheet of paper with the tasks that I asked them to perform with the tool on two networked computers. In part two the participants tried to work through their collaborative tailoring task. Part three consisted of a set of questions to the participants on how they experienced the work on their tailoring task, what they thought about the tool and certain parts of it, how they understood the sharing modes, and what they would suggest to improve.

In the first task that the participants had to work on, person A had to create a document template, then modify a toolbar and integrate a toolbar received from person B. Afterwards he had to save all of the above as document template connected with a toolbar in the private folder, and finally send it to person B. Person B had to create a toolbar with certain icons and send it to person A who then used it. The second task required A to define a group and then send a document template to the group, then change a toolbar and save it in the private folder and finally make this toolbar available in the public folder. In this task person B had to copy the toolbar from the public to the private folder and then load it via the preview mode.

The tasks required some coordination between the participants: they had to decide who was to do what in which order. By observing their discussion about how to proceed I gathered a first insight in how they perceived the tool and its affordances. The idea of having people discuss how they might proceed to reach a common goal is part of *constructive interaction* (see Kahler 2000). This method is particularly suitable for the CSCW context since for collaborative work talking with your colleagues about how you plan to achieve things is very natural. Insofar, constructive interaction lacks the awkwardness that accompanies the *thinking aloud* method where participants utter what they think while evaluating a computer system (Nielsen 1993). Constructive interaction was also used after the initial phase of coordination between the participants when they worked on their task. Thus, I was able to log the comments, (mis)understandings and perceptions related to the common work on the tasks.

The usability test resulted in findings on different levels. Most obvious, there were some shortcomings of the interface. Some buttons created misunderstandings and needed to be renamed. One button's name needed to be changed from "delete" to "deactivate" since the action that it triggered was hiding a toolbar. Another button needed to be renamed from "copy" to "adopt" where participants could decide if they wanted to move an adaptation that was sent to them to their private folder.

Moreover, it became clear that there was a need to be able to delete an adaptation from within Microsoft Word 97 rather than having to use the file manager. This also resulted in the suggestion to introduce the role of an administrator as a person who is allowed to delete adaptations in the public folder.

All of the participants considered the possibility to save, connect and distribute adaptations to be very helpful for their work. Although not all participants were expert users they were all able to use the tailoring functionality and the sharing functionality. The overall usability of the tool was perceived to be good. One participant explicitly said that he would tailor his word processor more in the future since he now knew how to do it and was no longer afraid that the tailoring activities would make the software unusable. This was mainly due to the preview mode. The two participants who were network administrator and experienced user said that such a distribution of adaptations would be very helpful for their organizations. The discussion following the tasks revealed that the

participants' conceptual model of how the distribution of files worked was very close to how we, the designers, had intended and implemented the distribution. This is an important result insofar as often and particularly in a more complex group work setting a misperception of the underlying model, e.g., about how links work or who gets to see and change what leads to inefficient usage or reduces a system's acceptance (Mark & Prinz 1997).

Quantitative Evaluation

Besides the qualitative usability test I also conducted a quantitative evaluation in which 32 persons participated. The aim of this quantitative evaluation was to find out how the adaptation browser and particularly its feature to send and receive adaptations performs in comparison to the sending mechanism already implemented in Microsoft Word 97 in the file menu. The menu item *Send To* spawns an external email client with an outgoing mail that contains the current Microsoft Word 97 document template as attachment. My hypothesis was that the adaptation browser would not rank worse than the internal mailing mechanism even if it was unknown to users.

To test this hypothesis 32 persons of at least average computer skills had to test both the adaptation browser and the internal mailing mechanism. On a 1 to 3 scale on how often they work on a computer (never – sometimes – often) they averaged a 2.56; on a 1 to 3 scale on how often they use email (never – sometimes – often) they averaged a 2.44; 29 of them used Microsoft Word as a word processor. Their task was to get to know both ways of sending and receiving and in a third step to decide which they like best and to send and receive a file in this preferred way. Both ways of sending took place directly from Microsoft Word 97. Receiving files with the adaptation browser could be done directly from Microsoft Word 97, the other way to receive files was via ordinary email. The files then had to be loaded to Microsoft Word 97. Performing the task took them from 9 to 26 minutes with an average of 15 minutes and 19 seconds. Of the 32 persons in the test 14 (44%) preferred the adaptation browser for both sending and receiving, 11 (34%) preferred the internal mailing mechanism for both sending and receiving, 3 (9%) preferred the adaptation browser for receiving but the internal mailing mechanism for sending, 1 (3%) preferred the adaptation browser for sending but the internal mailing mechanism for receiving, and 3 (9%) used the internal email mechanism for sending but did

not receive a file due to time constraints in the third part of the task. After the test I asked the participants how they liked the adaptation browser and the internal mailing mechanism on a scale from 1 to 6 (very bad – bad – rather bad – rather good – good – very good). They gave eight marks according to the combination of the dyads adaptation browser or internal mailing mechanism, sending or receiving, functionality or usability. The following means resulted from the participants' judgements:

adaptation browser	functionality	sending	4.9
internal mail mechanism	functionality	sending	4.8
adaptation browser	functionality	receiving	4.6
internal mail mechanism	functionality	receiving	4.7
adaptation browser	usability	sending	4.7
internal mail mechanism	usability	sending	4.5
adaptation browser	usability	receiving	4.5
internal mail mechanism	usability	receiving	4.5

All of the means are between 4.5 and 4.9 with a maximal difference of 0.2 between the adaptation browser and the internal mail mechanism in any given category. The results show no significant difference for the adaptation browser and the internal mail mechanism which proved the hypothesis that the adaptation browser would not rank worse than the internal mailing mechanism even if it was unknown to users. Despite the fact that the adaptation browser was only an unoptimized prototype with the first version of the user interface the participants could obviously detect the value in the strong integration and the enhanced functionality of the adaptation browser.

Conclusion

While the fact is well known for quite some time that tailoring activities are often carried out collaboratively, there is a lack of support for this. Based on an empirical study, four different use situations were presented about how joint tailoring of a word processor takes place. Up to now generic single user applications – like word processors – do not provide support to share adaptations among its users. Nevertheless, with the increasing number of

computer networks, a technical infrastructure to share such artifacts is often existing. To clarify how support for joint tailoring of generic single user applications could look like, the functionality of Microsoft Word 97 was extended. Based on the requirements derived from the above use situations the functionality provides a public and a private repository for adaptations as well as a mailing function. It is fully integrated into the user interface of the word processor. Finally, the results of a usability test were presented, which indicates that even non-expert users understood the concepts. Moreover these results hint to the fact that such a tool may increase the frequency of tailoring activities.

I assume that such a tool may also serve as a medium that encourages groups to discuss group standards, e.g., for letter templates that then can be shared. The systematization of customizations (Trigg & Bødker 1994) resulting from a collaborative tailoring process would then contribute to common norms and conventions needed for collaborative work (Wulf 1997).

Suggestions for the use of such a tool cannot be restricted to technical design requirements but must include organizational suggestions as well. I am convinced that the establishment of a "gardener" (Nardi 1993) or "translator" (Mackay 1990), e.g., a local expert responsible for the coordination of tailoring activities is a vital part of tailoring measures in organizations.

Right now it seems that adaptations are most usefully applied in the organizational context of their emergence supporting the tasks they are made for. The sharing tool in its current form is most helpful for small work groups with a rather similar work context. Future work will also address the question of technical and organizational scalability of such a tool. The hypothesis here is that the model of public and private spaces and the distinction between creator and user of the artifacts need to be enhanced to more than two levels when the group size exceeds a certain limit. Like in shared workspaces for general purpose, a more sophisticated access control model is needed (Pankoke & Syri 1997). Meta-information like annotations made by an adaptation's creator may help to compensate for part of a lacking context. Another enhancement of the tool would be to allow to distribute adaptations worldwide, e.g., via the World Wide Web (WWW). Thus, one could even think of supporting global teams or even establish widely accessible libraries for adaptations. Whether this is, however,

reasonable in the light of the poverty of organizational and task context is unclear. How context could possibly be provided and how large groups of participating contributors can be handled may be learned from recent experiences in distributed software development. This is particularly interesting when taking place without existence of a formal organization as in the case of the distributed development of Linux and its components.

However, in my experience it is clear, that collaborative tailoring does not scale easily. As always the question remains open how much administrative work the participating individuals are willing to contribute for the benefit of a group or organization and how much administrative effort is still reasonable to stay on the profitable side of collaborative tailoring. More refined tools to measure this and more refined categories to weigh the individual and group pains and gains against each other are needed.

Other plans for future work on the sharing tool include the integration of other tailoring functions of the word processor (e.g., macros) and the enhancement of applicability from Microsoft Word 97 to the whole Microsoft Office package. Besides adaptations, the tool can easily be extended to support the distribution for Microsoft Word 97 documents on which a group of people works.

Another alternative the time of which has yet to come is the embedding of such a mechanism to exchange Microsoft Word 97 related adaptations into a generic organizer of adaptations belonging to different applications. This organizer could combine mail mechanisms (or even be part of an email client) with the operating systems' functionality for access rights or shared workspaces and an enhanced explanation and commenting functionality. I believe that first steps in this direction are taken by work dealing with component architectures for CSCW and, in particular, for tailorability in groupware (see Stiemerling et al. 1999).

Acknowledgements

I thank my colleagues Oliver Stiemerling and Volker Wulf for many and ongoing discussions about collaborative tailoring. Jörg-Guido Hoepfner provided the implementation.

References

- Bentley, Richard; Dourish, Paul (1995): *Medium vs. Mechanism: Supporting Collaboration Through Customization*. In: Proceedings of ECSCW '95. Kluwer. pp. 133-148.
- Carter, Kathleen; Henderson, Austin (1990): *Tailoring Culture*. In: Reports on Computer Science and Mathematics no. 107, Åbo Akademi University 1990. Proceedings of 13th Information Systems Research Seminar in Scandinavia (IRIS). pp. 103-116.
- Cypher, Allen (1993): *The Practical Use of Macro Recording: A Case Study*. In: Proceedings of Human-Computer Interaction: Third International Conference, EWHCI'93. Springer-Verlag, Berlin. pp. 327-333.
- Haaks, Detlef (1991): *Anpaßbare Informationssysteme - Basis für aufgabenorientierte Systemgestaltung und Funktionalität*. In: Proceedings of Software-Ergonomie '91 - Benutzerorientierte Software-Entwicklung. Teubner. pp. 291-300.
- Henderson, Austin; Kyng, Morten (1991): *There's No Place Like Home: Continuing Design in Use*. In Greenbaum, Joan; Kyng, Morten: Design at Work - Cooperative Design of Computer Systems. Hillsdale, NJ, Lawrence Erlbaum Associates. pp. 219-240.
- JCSCW (2000): *JCSCW - Computer Supported Cooperative Work (CSCW): The Journal of Collaborative Computing*. Vol. 9 (1). Special Issue on Tailorable Systems and Cooperative Work.
- JCSCW (2001): *JCSCW - Computer Supported Cooperative Work (CSCW): The Journal of Collaborative Computing*. Vol. N.N. Special Issue on Evolving Use of Groupware (to appear).
- Kahler, Helge (2000): *Constructive Interaction and Collaborative Work: Introducing a Method for Testing Collaborative Systems*. In: acm interactions, Vol. VII (3 (May/June 2000)). pp. 27-34.
- Mackay, Wendy E. (1990): *Patterns of Sharing Customizable Software*. In: Proceedings of CSCW '90. pp. 209-221.
- Mackay, Wendy E. (1991): *Triggers and Barriers to Customizing Software*. In: Proceedings of CHI '91. pp. 153-160.

- MacLean, Allan; Carter, Kathleen; Lövsstrand, Lennart; Moran, Thomas (1990): *User-Tailorable Systems: Pressing the Issues with Buttons*. In: Proceedings of CHI 90. pp. 175-182.
- Mark, Gloria; Prinz, Wolfgang (1997): *The Establishment and Support of Conventions for Groupware*. In: Proceedings of INTERACT '97. Chapman & Hall. pp. 413-420.
- Mørch, Anders (1998): *Tailorable Groupware: Issues, Methods, and Architectures. Report of a Workshop held at GROUP '97, Phoenix, AZ, Nov 16th, 1997*. In: SIGCHI Bulletin, Vol. 30 (No. 2 (April 1998)). pp. 40-42.
- Nardi, Bonnie M. (1993): *A Small Matter of Programming*. Cambridge, Massachusetts, MIT Press.
- Nielsen, Jakob (1993): *Usability Engineering*. Boston, Academic Press.
- Oberquelle, Horst (1994): *Situationsbedingte und benutzerorientierte Anpaßbarkeit von Groupware*. In Hartmann, Anja; Herrmann, Thomas; Rohde, Markus; Wulf, Volker: *Menschengerechte Groupware - Software-ergonomische Gestaltung und partizipative Umsetzung*. Stuttgart, Teubner. pp. 31-50.
- Oppermann, Reinhard (1991): *Evaluation von adaptierbaren und adaptiven Leistungen im Tabellenkalkulationsprogramm EXCEL*. GMD. Arbeitspapiere der GMD 596.
- Orlikowski, Wanda (1996): *Evolving with Notes: Organizational Change around Groupware Technology*. In Ciborra, Claudio U.: *Groupware & Teamwork - Invisible Aid or Technical Hindrance*, Wiley. pp. 23-60.
- Paetau, Michael (1991): *Kooperative Konfiguration - Ein Konzept zur Systemanpassung an die Dynamik kooperativer Arbeit*. In: Proceedings of Computerunterstützte Gruppenarbeit (CSCW) '91. Teubner. pp. 137-151.
- Page, Stanley R.; Johnsgard, Todd J.; Uhl, Albert; Allen, C. Dennis (1996): *User Customization of a Word Processor*. In: Proceedings of CHI '96. pp. 340-346.
- Pankoke, Uta; Syri, Anja (1997): *Collaborative Workspaces for Time deferred Electronic Cooperation*. In: Proceedings of GROUP '97. pp. 187-196.

- Stallman, Richard (1981): *EMACS: The Extensible, Customizable, Self-Documenting Display Editor*. In: Proceedings of ACM SIGPLAN SIGOA. Massachusetts Institute of Technology. pp. 301-323.
- Stiemerling, Oliver; Hinken, Ralph; Cremers, Armin B. (1999): *Distributed Component-based Tailorability for CSCW Applications*. In: Proceedings of Proceedings of ISADS '99. IEEE Press. pp. 345-352.
- Stiemerling, Oliver; Kahler, Helge; Wulf, Volker (1997): *How to make software softer - designing tailorable applications*. In: Proceedings of DIS '97. pp. 365-376.
- Trigg, Randall; Bødker, Susanne (1994): *From Implementation to Design: Tailoring and the Emergence of Systematization in CSCW*. In: Proceedings of CSCW '94. pp. 45-54.
- Wasserschaff, Markus; Bentley, Richard (1997): *Supporting Cooperation through Customisation: The Tviews Approach*. In: Computer Supported Cooperative Work: The Journal of Collaborative Computing (JCSCW), Vol. 6 (4). pp. 305-325.
- Wulf, Volker (1997): *Storing and Retrieving Documents in a Shared Workspace: Experiences from the Political Administration*. In: Proceedings of Human Computer Interaction: INTERACT 97. Chapman & Hall. pp. 469-476.
- Wulf, Volker (1999): *Evolving Cooperation when Introducing Groupware-A Self-Organization Perspective*. In: Cybernetics and Human Knowing, Vol. 6 (2). pp. 55-75.

Fourth Paper

Constructive Interaction and Collaborative Work

Introducing a Method for Testing Collaborative Systems

Introduction

In trying to determine how people use interactive computer systems, many of us invest a considerable amount of time and energy to find out how users work and how they interact both with each other and with the computer. We want to know what they think about the interactive systems we provide for them and sometimes in cooperation with them and what we can do to make these systems fit their needs.

This article focuses on how the constructive interaction method helps system designers to determine whether the basic concepts underlying a system are well understood by users and whether its implementation, usability, and utility are satisfactory. We describe our experiences in using a form of constructive interaction to test a software package that supports a particular collaborative activity. The difference between our use of the constructive interaction method versus other models is that our test subjects use separate workstations in the same room to discuss their common tasks. Having thus adjusted the setting to the specific characteristics of computer supported collaborative work, we asked each person to carry out separate sets of predefined tasks that were linked.

Simply asking people whether they are satisfied with a newly introduced system does not suffice, because the reasons they give may not reflect their actual views or behavior (Cicourel 1964). To avoid the shortcomings of such a straightforward approach, the thinking aloud method has been used in order to both gain a more adequate understanding of how a person views a system and test the system's usability (Nielsen 1993). In a standard

thinking aloud test, a person has to work on a predefined task while continuously verbalizing her or his thoughts. This method yields a set of verbal utterances combined with actions about the task. The behavior of the person tested can be audio- or videotaped, and analyzing several of those tests may reveal how people understand or misunderstand the computer system and how to reduce misunderstanding. However, this method clearly has drawbacks. First, interaction is limited because the user mainly reports his or her experiences to the researcher. Second, the setting may seem unnatural to many people and make them feel they are being observed. And third, the researcher might interact too much with the person tested and bias the result.

To circumvent the difficulties of usability testing involved in the thinking aloud test, the constructive interaction method is carried out using two subjects. The two subjects are asked to perform a task together, which usually leads to arguments about what to do next and how to do it and explanations to each other of why they did what they did. Since this type of interaction is more natural than that in the thinking aloud test, and since interaction between researcher and tested individuals is minimal, the results can be considered to be of a relatively high ecological validity.

This method of observing two people in the solution of a common task in order to better understand learning processes, mental models, and aspects of a system's usability has been labeled with different names, depending on the focus of the researchers involved. Miyake (1986), for instance, calls the method "constructive interaction", whereas Kennedy (1989), who applied the method in the context of usability testing, refers to it as "co-discovery learning." When applied to usability testing, other names have been introduced, such as "paired user testing" (Wildman 1995) or "co-participation" (Wilson 1998). It should be noted, however, that the last two terms usually imply an environment in which two people work together at the *same* workstation, whereas the form of constructive interaction presented in this article requires two people to perform a collaborative task on *two separate* workstations. To differentiate between these two approaches, I will use the term "constructive interaction for testing collaborative systems" (CITeCS) for the setting with two separate workstations (see table 1 for an overview of the different forms of constructive interaction).

Aspects of Constructive Interaction

The concept of constructive interaction was introduced by Naomi Miyake (1982, 1986), who asked of test subjects to discuss and solve a problem - in her case, how a sewing machine works. Miyake was interested in the iterative process of understanding that takes place when people discuss a problem and pass through several levels of understanding. Her study proved the existence of consecutive levels of understanding the sequence of which followed a certain pattern. Miyake also showed that having a pair of individuals discuss a topic and work collaboratively on a solution revealed much about their underlying assumptions, mental models, and understanding of the topic.

O'Malley et al. (1984) were the first to explore the potential of constructive interaction for human-computer interaction and the conditions under which it might be effective. They conducted two studies, each of which involved two participants. The first study was a tutorial session in which an experienced user introduced a novice to a system. The session revealed several sources of confusion for the novice. In the second study, two people were asked to find out how a particular command interpreter worked. They discussed possible strategies and tried out various aspects of the system's functions to support their points of view.

Mayes et al. (1990) used constructive interaction by asking pairs of subjects to make collaborative decisions about how to proceed through a hypertext. In their study, the authors drew conclusions about the lack of benefits of hypertext learning systems relative to human-computer interaction and reported evidence that constructive interaction itself can promote learning.

Kennedy (1989) was the first to describe constructive interaction as a usability testing method in a commercial setting at Bell-Northern Research. Since then, constructive interaction, improved and modified in various ways, has been explicitly and widely used in usability testing.

The main advantage of constructive interaction is that it yields a rich set of qualitative data that provide valuable insight into how people perceive situations, how they go about solving problems, and, in particular, how they perceive the conceptual framework and usability of a given system. Sasse (1996) suggests that constructive interaction is particularly well suited for exploratory studies. A study by O'Malley et al. (1984) revealed that

constructive interaction can be quite useful for exploring users' understanding of system concepts. The reason is that differences of opinion lead test subjects to articulate the rationale behind their hypotheses, thereby enabling the observer to understand how the subjects perceive the system. Mayes et al. (1990) argue that constructive interaction differs from many other methods because it does not aim at reducing data but rather at exposing as much of the underlying cognition as possible. According to Wildman (1995), constructive interaction is a good method for early usability testing when the design process focuses on general issues of navigation, representation, organization, and functionality. Kennedy (1989) reports that video recordings of experiments involving constructive interaction provide more interesting, informative, and convincing data than video material from thinking aloud sessions. Kennedy also used the method in her interaction with developers. Seeing users interact in a video about their trouble with using the product was much more convincing than detailed descriptions of usability test results and statistics.

One drawback of such an approach is that the abundance of data cannot be easily evaluated quantitatively. If you want to go beyond purely qualitative statements and perform detailed error analyses or compare different pairings, the data must be carefully transcribed and analyzed. Given the richness of the information, this is likely to be time consuming.

An important issue in constructive interaction is the relationship between the individuals paired. Often it is reasonable to have two individuals who have the same level of knowledge or expertise and whose communication will therefore be marked by an exchange of opinions on how to work on a task. Sometimes, however, it might be helpful to choose individuals with different levels of knowledge in order to create a situation in which the interaction is guided by one person. However, differences in expertise or verbal style (e.g. outspoken or talkative versus shy or restrained) or a hierarchical relationship between the individuals may hamper feedback. Wilson (1998) cites positive experiences with recruiting two individuals as a pair, for example, by asking willing participant if she or he would like to bring along someone to do the test with.

Several suggestions have been made to increase the number of individuals involved in constructive interaction. Westerink et al. (1994) have proposed settings in which three people have to interact with each other. In such a situation, the two people taking the usability test were asked afterwards to

describe their experience to a *listener*, whose task was to elicit a summary of their impressions. Wilson (1998) reports an interesting case in which two system administrators and two users took part in a session during which the administrators explained the product to the users.

Constructive Interaction for Testing Collaborative Systems

I began to explore the topic of constructive interaction when my colleagues and I researched tailorable systems and collaborative work. In our research, we investigated how collaborative tailoring of off-the-shelf applications can be supported by technical mechanisms.

Setting

System

In the study we conducted, we decided to work with a word processor, because it is a good example of a widely and extensively used software. In order to learn more about how groups of users tailor their tasks collaboratively, we carried out a field study at four different organizations. The study yielded a number of different collaborative tailoring scenarios, all of which focused on the exchange of document templates and toolbars, that is, graphical representations of functionality. By analyzing these scenarios, we developed requirements for the design of a tool to be used as an add-in to the word processor, or rather, an extension of its functionality using the programming interface. This add-in, henceforth called *the tool*, provided some functionality for collaborative tailoring, that is, for sharing and distributing changes to the functionality or appearance of the word processor or document templates that could then be used or modified by other people (see Kahler & Stiernerling 1999 for information on tailoring and the tool).

The basic functions included loading and saving document templates and toolbars. It was also possible to combine a document template and several toolbars in a package in order to support specific word processing tasks, such as the design of a Web page or the writing of a mathematical paper. The collaborative aspect was added by the functionality we provided for sharing document templates and toolbars between the creator and other persons by both a sending and an access mode. In order to support centrally

administrated environments, adaptations could be sent to groups of end users. The access mode allowed users to simply store the tailored artifacts in a shared workspace. If another user were searching for a certain adaptation she could access the required templates or toolbars in this shared workspace, the *public folder*.

Method

To test the tool, constructive interaction was an obvious choice because we wanted to have pairs of users perform tasks collaboratively. The test had two goals. On the one hand, we wanted to find out if and to what extent the users taking part in the test understood the concept of sharing tailored artifacts and how it was implemented in the tool. On the other hand, we expected the experiment to yield clues for improving the usability and utility of the tool.

Enhancing the constructive interaction setting as outlined previously, we set up two workstations, which the two test subjects were to use to perform their common task. This setting reflects the nature of asynchronous distributed work, whereby two individuals, A and B, take turns in performing a sequence of actions. Unlike most computer supported collaborative work situations, however, the two individuals were located side by side in the same room so that they could talk to each other face-to-face and so that each was able to see what was happening on the other's monitor. All test subjects had participated in the field study. The subjects teamed up in each of the two pairs knew each other but had not worked closely together before. The tests took place in one of our offices.

The test subjects had to work collaboratively on two tasks, each of which involved several subtasks. The tasks consisted of jointly creating and refining a word processor's document template, including a toolbar. Before testing the individuals, we explained to them the basic functions of the tool and the aim of the experiment, which was to test the tool's usability and utility. In a first step, the task of Person A was to create a document template, modify a toolbar, and incorporate another toolbar that she received from Person B. Afterwards, she had to save all of these elements in a document template connected with a toolbar in her private folder and send it to Person B. Person B, in turn, had to create a toolbar with specific icons and send it to Person A for further usage. The second task required Person A to define a group and send a document template to the group. She then had

to change a toolbar, save it in the private folder, and make the toolbar available in the public folder. In this phase of the test, Person B had to copy the toolbar from the public folder to his private folder and then load it using the preview mode. Both participants had the same written task description, which was divided into two sections, “Tasks for Ms. A” and “Tasks for Mr. B.” From the task description they could see when it was their turn to interact with their workstation. We encouraged the test subjects to read the task description and to check with each other whether they knew what to do; they were also encouraged to discuss the next step that each had to take in the course of the task. Work on these tasks lasted about 30 minutes.

A researcher and the developer of the tool were present to observe each of the test pairs. Both took notes during the tests. Moreover, the test was audiotaped to support those notes in cases of doubt and to be able to extract quotes. After the test we reviewed a brief questionnaire concerning aspects of working with the tool that could not be dealt with in the tasks.

Results

System

The results of our constructive interaction sessions concern different levels. First of all, the test showed clearly that the interface of the tool needed to be improved. Some buttons caused misunderstandings and had to be renamed. The name of one button, for instance, had to be changed from **Delete** to **Deactivate** because its function was to hide a toolbar. Another button, which had originally been labeled **Copy** and allowed users to move a tailored artifact that was sent to them from the inbox to their private folder, was renamed **Adopt**. Moreover, it became clear that users should also be able to delete a tailored artifact from within the word processor rather than having to use the file manager. This modification also resulted in a proposal to introduce an administrator, who would be allowed to delete tailored artifacts in the public folder. All of the participants considered it an advantage to have the possibility to save, combine, and distribute tailored artifacts. Although not all test subjects were expert users they were all able to use the tailoring function and the sharing function. The users perceived the overall usability of the tool to be good.

Two participants, one a network administrator and the other an experienced user of the particular word processor being used, said that such a

distribution of tailored artifacts would be quite helpful for their organizations. The constructive interaction sessions revealed that the participants' conceptual model of how the distribution of files worked was close to how we, the designers, had intended and carried out the distribution. This is an important result insofar as a misperception of the underlying model (for example, about how links work or who can see and change which elements) often leads to a user's inefficient use or lack of acceptance of the system. This holds in particular for more complex work group settings.

Method

We found that, for our purposes, constructive interaction for testing collaborative systems proved to be effective. The topic of collaborative work that this version of constructive interaction focused on was, for the first time, connected to the method. The tests showed that our enhancement of constructive interaction methodologically suited the questions raised by computer supported collaboration. The collaborative nature of the task and the fact that the system was a medium for the test subjects' collaboration made CITECS the method of choice.

Constructive interaction as it was employed in earlier studies was a useful framework to start developing ideas about testing a collaborative task, because it already involved communication between two people working on a common task. The new aspect that we added with our enhanced model is the distribution of parts of the task among the two people. Introducing a second workstation, while still allowing face-to-face communication, combines the advantages and the natural communicative setting of constructive interaction with the main features of collaborative work. In our tests, this approach resulted in lively discussions among the participants, which provided valuable insights into the problem-solving process as well as into the interests of the partners and the different roles they assumed in their collaboration. Our setting proved to be well chosen because either user in a test pair could ask the other person what impact their actions would have on the other person's work. This approach made it possible for each user to understand *both* sides of the collaborative process. Its benefit was thus twofold: (1) it helped test users to understand the system and (2) it allowed us to gain a number of interesting insights into how the users'

perspectives on their particular part of the common task was influenced by our tool and how our design influenced their collaboration.

Aside from the quite awkward option of employing one tester as a dummy user, an alternative setting for this kind of task would be to let the two users work in separate rooms, each of them observed by a tester. However, such an approach not only would involve more resources but, in order to create a more realistic collaborative environment, would unnecessarily hamper the communication between the test subjects. Using two workstations with distinct but strongly connected tasks for the two subjects, as we did, instead of one workstation with two virtual screens, which would have been another alternative, prevented one of the paired individuals from assuming a dominant role.

Although the setting with two workstations in one room proved particularly useful for examining collaborative work, several aspects of our test can be related to findings of other researchers who employed constructive interaction or paired-user testing. Like others before us, we found that having two users discuss and perform a common task was a useful means for understanding the users' perception of the system concepts and for uncovering usability flaws. Compared with thinking aloud sessions, which we had used previously to explore other issues, the discussion between the two test subjects seemed much more "real" than the utterances from thinking aloud test takers.

For both the implementation of the tests and the evaluation of the data, we chose a simple setting that required neither laboratory nor sophisticated video equipment. Furthermore, we did not transcribe the tapes in detail or perform a quantitative evaluation because we felt that the extra work involved would have been disproportionate to the potential benefits for our research goal. The richness of the data shows that such a technologically modest approach can be useful in academic or other settings where resources are limited. This modest approach also has the advantage of providing a more natural setting for testing collaborative activities because it can be carried out at people's work places where individuals feel more comfortable than in the artificial "workplace" constructed for them at a research institute.

Our CITeCS activities were connected to other forms of learning about the usability of our tool in two ways: before the test, we conducted individual

interviews about tailoring software; after the test subjects completed a questionnaire in which they were asked, for instance, to draw a map of where they thought certain artifacts were located at different times in the process. Both of these additional techniques proved useful. The test subjects needed fewer introductory explanations and were able to understand the rather complex task because they had already been acquainted with the topic. The questionnaire complemented the test results and supplied us with insights that the CITECS method could not have provided.

Prospects

Our experience with constructive interaction has encouraged us in several ways. In times of increased computer supported collaboration over distance, CITECS offers possibilities for testing over distance with two or more test takers connected by audio or video, or both, using a collaborative system and performing a set of collaborative tasks. Furthermore, constructive interaction is not limited to testing purposes; it can also be used for a hybrid that combines training users with fine-tuning a system to the users' specific needs. In the POLITEAM project (Pipek & Wulf 1999), we customized a system to the needs of a group, introduced it, and trained the users. Using constructive interaction by pairing or grouping persons in the training sessions and having them perform tasks in such a situation could serve two purposes. First, it would be an appropriate way of teaching them the basics of the system that has been customized to their needs to the best of our knowledge and helping them to understand the specific aspects of collaborative work and the interrelations of the actions that group members perform with the system. Second, we, as system designers or people who customize systems for others, could learn both about characteristics of the group that we may not have foreseen or fully understood and about the specific requirements for fine tuning the system.

We are convinced that there is still more potential in CITECS, and we will continue to improve and use it to design, introduce, use, and evaluate collaborative systems.

References

Cicourel, Aaron V. (1964): *Method and Measurement in Sociology*. New York, Free Press.

- Kahler, Helge; Stiernerling, Oliver (1999): *"Pass me the toolbar, please!" - Cooperative Tailoring of a Word Processor*. In: Workshop on Implementing Tailorability in Groupware at the WACC '99 organized by Teege, Gunnar; Kahler, Helge; Koch, Michael; Stiernerling, Oliver. <http://www11.informatik.tu-muenchen.de/workshops/wacc99-ws-impltailor/>.
- Kennedy, Sue (1989): *Using Video in the BNR Usability Lab*. In: SIGCHI Bulletin October 1989, Vol. 21 (2). pp. 92-95.
- Mayes, Terry; Kibby, Mike; Anderson, Tony (1990): *Learning about Learning from Hypertext*. In: Jonassen, David H.; Mandl, Heinz: *Designing Hypertext/Hypermedia for Learning*. Berlin etc., Springer. pp. 227-250.
- Miyake, Naomi (1982): *Constructive interaction*. University of California. Center for Human Information Processing: CHIP-report 113.
- Miyake, Naomi (1986): *Constructive Interaction and the Iterative Process of Understanding*. In: *Cognitive Science*, Vol. 10 (2). pp. 151-177.
- Nielsen, Jakob (1993): *Usability Engineering*. Boston, Academic Press.
- O'Malley, Claire E.; Draper, Steve W.; Riley, Mary S. (1984): *Constructive Interaction: A Method for Studying Human-Computer Interaction*. In: *Proceedings of Human-Computer Interaction - INTERACT '84*. Elsevier. pp. 269-274.
- Pipek, Volkmar; Wulf, Volker (1999): *A Groupware's Life*. In: *Proceedings of ECSCW '99*. Kluwer. pp. 199-218.
- Sasse, Angela (1996): *Eliciting and Describing Users' Models of Computer Systems*. School of Computer Science, The University of Birmingham. Unpublished Ph.D. Thesis.
- Westerink, J. H. D. M.; Rankin, P. J.; Majoor, G. M. M.; Moore, P. S. (1994): *A New Technique for Early User Evaluation of Entertainment Product Interfaces*. In: *Proceedings of Human Factors and Ergonomics Society 38th Annual Meeting*. pp. 992.
- Wildman, Daniel (1995): *Getting the Most from Paired-User Testing*. In: *interactions*, Vol. 2 (3). pp. 21-27.
- Wilson, Chauncey (1998): *Pros and Cons of Co-Participation in Usability Studies*. In: *Usability Interface*, Vol. 4 (4).

Resear- chers	Motivation	Number of Work- stations to Perform Task	Recor- ding Tech- niques	Data Analysis	Embedment	Results Concerning Method	Variations
Miyake	Interest in the iterative process of understanding that takes place when people discuss a problem and pass through several levels of understanding	0	Video-tape, audio-tape	Transcription, double protocol coding, categorization of actions, marking breaks in utterances	Subjects watched tape with experimenter to help clarify issues	Constructive Interaction suitable to reveal iterative process of understanding	None
O'Malley et al.	Explore the potential of constructive interaction for human-computer interaction	1	Not described	Transcription, diagram of session showing main topics	Not described	Subjects show different points of view about common problem Emphasis should be on understanding or developing concepts, as opposed to learning procedures	Mayes et al.: using constructive interaction to explore learning from hypertext
Wildman	Usability testing	1	Video-tape, logging of activities	Analysis facilitated by extra workstation for test subjects to take notes	Not described	Paired-user testing low-cost collection of data, users' reasoning processes become visible, friendly ambience of users solving problems together, increased testing time for increased data volume and reduced post-test transcription	Kennedy: usability testing of telephones, laborious quantitative analysis, post-test interviews Wilson: collection of expert statements about co-participation Westerink: after the test the pairs are asked to describe their experience to a third person
Kahler – CTeCS	Test system for collaborative work	2	Taking notes, audio-tape	Qualitative analysis	Pre: field study with interviews Post: question-naire after test	Two workstations in one room well suited for testing collaborative systems and distributed tasks Previous results about method confirmed	None

Table 1. Constructive Interaction and Collaborative Work

Fifth Paper

Developing Groupware with Evolution and Participation - A Case Study

Abstract

This paper is about experiences with the evolutionary and participatory development of a search tool for a groupware system. After the description of different software engineering approaches and their use for evolutionary and participatory software development the POLITEAM groupware project is presented. The procedure of how the search tool for POLITEAM was developed including interviews, workshops and the usage and evaluation of prototypes is described. The resulting search tool is presented. The paper concludes with remarks about the usage of participatory design methods for the introduction and customization of generic groupware in different organizational settings.

Introduction

Approaches to Software Design

For a long time the development of software applications was mainly technically determined. The top-down waterfall model of the software life cycle (cf. Boehm 1976) and revised versions of it became the standard for software development. While this model proved to be appropriate for some classes of software, it didn't work well with others. Particularly for the development of "embedded programs" (Lehman & Belady 1985) that are characterized by the interdependence between the software and its environment the waterfall model proved to be inadequate. Several software engineering approaches and software life cycle models have been developed to overcome these shortcomings that give more consideration to the organizational environment of the program-to-be. Among those are

Boehm's spiral model, Henderson-Sellers' object-oriented fountain model, Hesse's EOS model, and Floyd's STEPS model.

In Boehm's risk-driven spiral model (Boehm 1988) several cycles are involved each of which includes the planning of the next phase, determining objectives and constraints, evaluating alternatives and resolving risks, and developing the next-level product. With the emergence of object-oriented programming, analysis and design Henderson-Sellers and Edwards (1990) proposed their fountain model for the object oriented life cycle. It is based on the iteration and overlapping of consecutive phases (e.g. system design, program design and coding) and on overcoming the need to freeze specification at an early stage by using autonomous classes that can easily be modified without having strong side effects on other parts of the system. Another approach involving object orientation is Hesse's EOS model (Hesse & Weltz 1994). It is based on merging evolutionary system development with the principles of object orientation. Analysis, design, implementation and application are considered to be the four activities of a software development cycle that are performed on the system-, component-, and class-level with increasing frequency. The EOS model is explicitly based on the idea that software projects create technical artifacts while shaping the structure of work in a particular organization, thus dismissing the notion of software development as a mere engineering process.

All of these approaches stress the importance of the organizational environment for software development with the overall notion that the design of software should be worked on beyond the early stages of a software's life cycle but must contain evolutionary aspects that allow for design changes and adaptations during software development.

Floyd's STEPS model of software development (Floyd et al. 1989a) explicitly introduces a new aspect into software development for embedded programs. It is strongly inspired by the Scandinavian approach (cf. e.g. Floyd et al. 1989b, Greenbaum & Kyng 1991, Ehn 1993) to system design with its stress on user participation (also cf. Floyd 1993). Incorporating strong user participation STEPS bridges the gap between software engineering and the discussions about participative software design lead in the Participatory Design (PD) and Human-Computer-Interaction (HCI) communities. STEPS is meant to develop embedded programs not only in an evolutionary process but with users playing a decisive role in the development process. Software development is seen as a process of mutual

learning where the developers contribute their knowledge of formal methods and software development and the users contribute their knowledge of the work domain. In the STEPS model each of them have tasks in the development process with some of the tasks being common (see figure 1).

Developing Groupware

The question of how to develop software that is strongly embedded in the organizational environment is particularly important for CSCW (Computer Supported Cooperative Work) research. Here, a group's particular ways of communicating and cooperating need to be supported. These can be vastly different between different groups and might also change within one group in the course of time. In order to be able to develop adequate software to support such a group it is necessary to find out the group's needs and then develop or adjust the software accordingly. This should be done in a process that includes both participation and evolution. Participation of members of the work group gives them the chance to put in their work and group experiences while evolutionary development of the software is necessary since it is hardly possible to meet the software needs of a dynamic system like a work group with a software right away and without adjusting the software along with the experiences made in the work group.

Although it was originally not made up for the development of groupware the intriguing aspect about the STEPS model is that it combines user participation in different parts of the process with a cyclic approach allowing for stepwise improvements of the existing prototype or program version. Thus, the particular difficulties of developing an embedded program can be faced in an appropriate way. User participation in the design phase helps to understand the structure of work and the particular needs of an organization or a group of users while the cyclic evolution of the program is bridging the gap between specification and usage by having the software gradually approximate to the current work practice. Considering the growing environmental dynamics and complexity organizations have to deal with and the emergence of post-tayloristic forms of organization more and more programs will be strongly embedded in organizational settings and will need to be developed accordingly. Some authors have remarked that STEPS has only little focus on the actual participatory activity and does not involve exploratory prototyping (Grønbaek et al. 1995). While this is true as far as explicit statements go, STEPS provides a good base to work on and

needs to be filled with concrete actions when working on system development.

So, being based on the idea that software development should be an evolutionary and participatory process the STEPS model can be considered to be a good start for evolutionary and participative development of groupware with all its special aspects to be taken into account. This is why it was decided to use STEPS in the POLITEAM project.

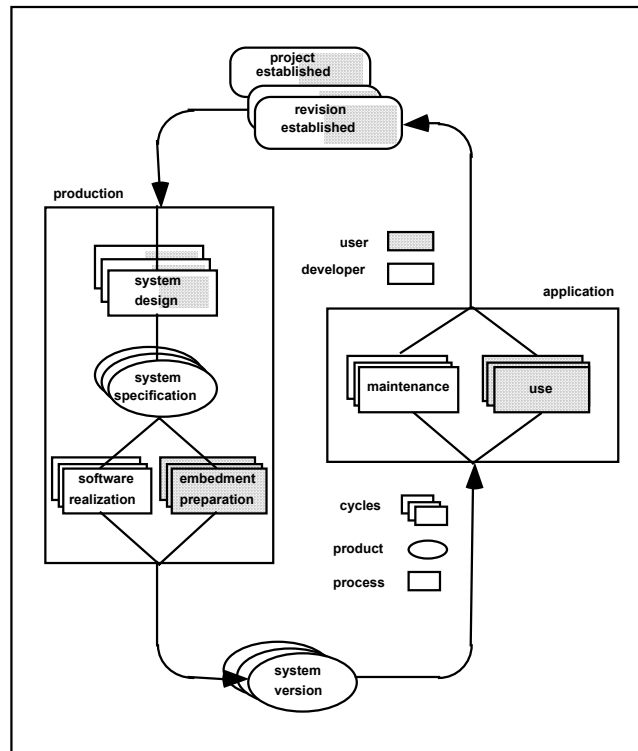


Figure 1: STEPS model for software development (Floyd et al. 1989)

The POLITEAM Project

In 1989 the wall between East and West Germany came down. This resulted in many social, economic, and political changes one of which was the decision that Germany's capital was to move from Bonn to Berlin. Since the movement of such a big administrative organization with some thousand employees could only be done stepwise and since it was decided that some of the German federal administration was to remain in Bonn the government

faced the need to come up with ideas to support the now geographically distributed government. Different parts of the government that were only miles apart in Bonn were to be partly in Bonn and partly in Berlin with a distance of about 400 miles. Among other activities the government set up the POLIKOM program to support research and development of adequate ways and tools for telecooperative work.

Taking part in this program is the POLITEAM project consisting of industrial partners (VW-Gedas as software company), research institutes (University of Bonn and GMD, the German National Research Center for Information Technology) and application partners from the federal administration, a state administration and the software engineering department of a car manufacturer. The aim of the POLITEAM project is to develop a system to support distributed work in large organizations. This is done by providing a workflow component to handle circulation folders that structure the workflow and by implementing the metaphor of a “shared desk” that integrates document processing tools. This means that the users of the POLITEAM application work on a desktop where they can place objects that others have access to, e. g. shared folders or text objects that are editable by a group of persons (cf. Klöckner et al. 1995).

POLITEAM is based on Digital’s LinkWorks™. The functionality of LinkWorks™ is used, enhanced, and changed by adding software components and using the LinkWorks™ application programming interface. POLITEAM is a client/ server application where usually each client provides document processing applications (e. g. Word for Windows) while the server stores the documents and meta-information like access rights, a list of persons who are to receive a circulation folder, or the position of objects on one’s desktop. The design approach of POLITEAM explicitly emphasizes evolutionary and participative aspects and is based on Floyd’s STEPS. For each of the application partners that were to introduce POLITEAM into their organization their work and organizational structure was analyzed. After configuring the first versions of POLITEAM to each of the application partner’s needs it was introduced in their organizations so that about 40 persons altogether work with the system right now. In the course of the project more users will be provided with POLITEAM. The introduction was accompanied by training the users to work with the system and after that the application partners were visited regularly by user advocates (cf. Mambrey et al. 1996), i. e. every week or fortnight, to give feedback about their experiences with the system and to suggest improvements for the upcoming

next version of POLiTEAM. Learning from these visits and workshops that were held with the application partners the current POLiTEAM version will be reshaped to better meet the application partner's needs.

The following chapter provides an example of how user involvement resulted in system evolution for a tool from the POLiTEAM system.

Developing a Search Tool - Experiences

Existing Search Tool

The basic version of LinkWorks™ had a tool implemented that allowed for searching objects. With this search tool one could basically find any object known to the system. The search tool provided different search criteria for an object such as the name, the object class (e.g. "text" or "folder"), the date of its last change, the name of its creator and more. To protect the privacy of the workgroup's members the possibilities of the search tool had to be restricted by providing objects with a search flag that marks if an object can be found by the search tool. This flag cannot be set directly by the creator of an object but only via an access profile containing the information that this object is unsearchable.

With the application partners we agreed on three different access profiles that should be configured and provided for them with the option of refining the access profiles later (e.g. by allowing or prohibiting the attachment of an object to an e-mail) and thus increase the number of access profiles. The most general of the three initial access profiles for an object was "public" where every person is allowed to see / read and change / write the object. The second access profile was "for your information" meaning that the object could only be read but not changed by anyone but the creator and the most restrictive access profile was "private" where no one but the creator of an object could read or write it. Of these three access profiles "private" was the only one where the search flag of the object was not set so that this object was unsearchable, i.e. not visible for the search tool. By allowing for granting the "private" access right to objects and thus preventing them from being found by the search tool basic issues of privacy were ensured.

Still the search tool was expected to make problems in the daily work of the application partners so it was finally decided not to use the existing search tool at the application partners' sites but to develop a new search tool that

should be more adequate to the users' needs. To understand the problems that arose with the original search tool some more of its functionality must be explained.

To support cooperative work on a document (e.g. text) LinkWorks™ provides three possibilities. The users can either work on one electronic document that is treated like a real world paper document. In this case there is only one copy of the document that can be worked on by one person at a time and that has to be moved to and fro for different persons to see or change it. The second way for cooperation is to make one or more copies of an existing document that are treated like real world copies, i.e. that can be worked on independently. If the aim is to produce a single document of these copies they must be merged manually. The third possibility is encouraged by LinkWorks™ and provides a way of handling a document that exceeds the possibilities of a paper document. Here, the document is shared between different persons in a way that they can all see this document on their desk at the same time. This is done by providing links from their desks to the document. If one person changes the document the links to the desks of others are immediately updated so they can see the changes. The advantage of sharing a document this way is that it is not necessary to send a document around for somebody else to change it or to send copies of a document around for others to be informed about the current state of the document. Moreover, working with links is more efficient than sending around copies that are worked on by different persons and that need to be merged afterwards.

Whenever the search tool was started it searched for objects in the system for that the specified criteria applied. So, if person A had created a text with the access profile "public" or "for your information" called "letter to J. Johnson" with a word processor and stored it in a folder on her LinkWorks™ desk then person B would find the text with the search tool request looking for all objects having the word "letter" in their name. Then the search tool would automatically create a link to this text and put it on B's desk in the "search" folder. The automatic creation of links by the search tool resulted in various problems concerning privacy aspects and data handling.

One problem consisted in the fact that person A was not informed about the fact that somebody searched her desk for an object and actually found one. Users working at the application partners' sites realizing that someone could

“snoop” on their desk which they considered a more or less private area they could feel uncomfortable about this. On the other side there is the need to search for objects in the system to get the information necessary to do the work. Moreover, for users it is extremely impractical to protect “their” objects from being found by giving them the access profile “private” since this would hamper shared editing of documents and cooperation in general.

Another problem caused the unintended deletion of files and was a major reason to decide for the redevelopment of the search tool. This unintended deletion resulted from the slightly inconsistent handling of files in the search window. The reason for this was that in the search window all objects found were represented as links to the original objects as described above. While in an “ordinary” window every deletion had to be confirmed, if someone pressed the delete key in the search window e.g. on a text found only the link in the search window was deleted without confirmation of the deletion and the object icon was removed from the search window but the original object still existed e.g. on someone else’s electronic desk. The same was true for found and deleted folders. This folder could contain linked and unlinked objects. The impression the users could get was that any deletion of an object started from the search window was harmless since only the link would be removed. This, unfortunately, was not true since when users opened the found and thus linked folder it contained objects that were not necessarily linked themselves. So when they would delete an unlinked object, say a text, in the found and linked folder it would be deleted for all other users that had this folder on their desk. This could lead to an unintended deletion of unlinked objects contained in a found folder.

A third incentive to work on the search tool was that the initial phase of internal use of the search tool made clear that the abundance of search criteria made it difficult to use the search tool. This resulted from the fact that the developers had implemented all criteria that could technically be searched rather than restricting the search criteria to a useful subset.

Redevelopment of Search Tool

The experiences from the initial phase of internal use concerning the search tool strongly implied that the search tool had to be redesigned and reimplemented in order to solve the existing problems with it. While so far the search tool had been just one of many features of POLITEAM the

experiences of the users had made it one of a few special things and problems to focus on.

In order to develop a search tool that supported the work for the application partners adequately the shortcomings of the existing search tool had to be overcome. We considered the aspects of searching that have to do with the particularities of group work to be of particular importance. So we decided to not only find work-arounds to deal with what had proved to be solved badly with the existing search tool but to go deeper and find out more about searching in a group and about the conflicts coming along with it. Our goal was to develop an improved search tool and learn more about potential conflicts and possible solutions that are relevant for people working with a groupware.

In the course of the redevelopment of the search tool different techniques of user participation and software evolution were involved. We conducted 10 interviews with interview partners from four application partner organizations, held four workshops where aspects of searching were raised, two of which were dedicated to search tool prototypes, and we developed three prototypes of search tools which were later evaluated.

These techniques were meant to bring up different aspects of requirements for the search tool and can be considered to be concretizations of the user-related activities in software engineering models involving user participation.

Interviews

To get a better understanding of how search in a work group is performed we started with conducting interviews about how people who cowork with each other search objects, i.e. documents, papers, or folders in an office environment. We talked to ten people, two of which worked in a library, two in a state administration which is an application partner, three in the office of a software company, and three in the office of a construction company. We deliberately chose interview partners that had worked and others that had not worked with POLITEAM to get input from a wide range of work practise and not be biased by users' previous experiences with POLITEAM. The interviews were led with one person at a time, lasted about 30-45 minutes each and were conducted along a questionnaire with 29 questions that served as a guide which left space for additional questions

and talk. The questionnaire consisted of open questions (answers in sentences, not just yes or no required), included physical and electronic search, and had two parts, the first of which related to the search activities of the interviewees in their offices (What are causes for a search? Describe how you go along? What tools do you use: telephone, post-it-notes etc.), while the second related to privacy issues. Here, the interviewees were to take the roles of both a person searching something in a work group and person 'being searched on', i.e. someone, who was asked about an object ('Do you know where this document is?') or whose room or desk or hard disk was searched by someone else (cf. Krüdenscheidt 1996). A similar role-oriented technique was used by Wulf & Hartmann (1994) researching on effects on visibility in a network.

The answers of the interviewees shed a light on different aspects of searching in a work group. Usually one of two problems is the starting point for a search, it is either the problem to find an object whose existence is known or the question if there is an object that contains the information searched. Three main causes for a search could be identified. These are the intention to work on a searched object (e.g. use components of an existing document to create a new one), the intention to gain information, and the intention to control something, e.g. the current state of a project, or someone. The objects searched were mainly internal (e.g. prepared speech for minister or inventory list) or external (e.g. legislative texts or offers from providers) text objects. The ways how and where objects are stored in a particular work place differed in the different organizations. This includes organizational as well as personal storage. Several personal preferences could be found which the interviewees stated to be efficient for themselves. On the organizational level we found different structures to sort and order documents like order by date, by internal or external order numbers or by task areas and within them again by project number and date. Moreover, in each of the four organizations a central place for the collection of documents exists, e.g. a registry in the state administration. The organizational search was often started by limiting the time range of the object to be found and by providing key words or restricting thematic areas if the document order structure supported this search. Interviewees in three of the four organizations worked on a computer and searched with the Microsoft Windows file manager or the word processor file manger. Here, the predominant search criteria are the file name, date, key words, and the author of a document.

The interviewees stated that they involved others in their search when they needed help, e.g. from a person in the registry who knew 'their' files or from a colleague who had worked with them on the document searched. Usually the others were not involved in the search process itself but by communication, i.e. they were contacted personally or on the telephone and asked questions about a document. For a search where others are affected the interviews showed a potential for conflict. The persons interviewed stated that usually the doors of their offices were open and that basically everyone could search in everybody else's room but that usually one wouldn't search in someone else's drawer but only on the desk and that this also depended on the relation of the persons. Potential conflicts showed where electronic search was discussed. Here, the symmetric design of the questionnaire allowed for every interviewee to take the role of a 'searcher' and the role of a person 'being searched on'. In the role of a person searching actively the interviewees pleaded for a nearly unlimited access for electronic search arguing that this would be helpful and necessary for cooperation and adequate for team work. When they took the role of a person affected by someone else's electronic search they felt uncomfortable knowing that everyone could look into their folders and considered this as an unwanted intrusion. One person (working with another system than POLITEAM) described her work practice where she would not move a document she worked on from her home directory that only she could access to a public directory until her work on the document was completed.

The interviews helped us to a deeper understanding of how people involved in team work search objects and they made clear that there was a particular need to handle the conflicts that might result from a search performed with a search tool on other person's electronic desks within POLITEAM.

Workshops Related to Group Work

Besides the interviews in this first step of the redevelopment of the search tool two workshops were held with eight users of the federal administration (ministry) application partner where searching was discussed among other topics. We incorporated workshops with a group of users in the development process since we felt they could bring out much more of the group dynamics than the interviews were able to.

At this time they had used POLITEAM for some while but they did not know the POLITEAM search tool which had been disabled before the system was

introduced there. In workshop I naming conventions for documents were discussed. The problem arose that in the office where documents were partly typed, processed and collected they used POLITEAM and DOS without POLITEAM and they were working with a very rigid name structure where document names had the DOS 8.3 form and where the first eight letters consisted of two letters for the document type (e.g. speech, letter, text from circulation folder) and the following six letters stood for the date. They did not want to change this rigid structure to stay compatible with the rest of the ministry. The people cooperatively working on the documents and writing the letters and speeches wanted to use POLITEAM's facilities for long (32 letters) names without sticking to the rigid conventions. This showed that the individual representation of information was important and that POLITEAM had to provide means to find objects that obeyed different naming or ordering criteria. The second workshop was held with the same group of users and served to introduce a new version of POLITEAM where it was possible to order the contents of a folder by different criteria like name, date, or key word. Also a viewer for a fast preview of documents and a facility for tree-like hierarchical representation of objects in POLITEAM were presented. The users said that these three features would be of great help in finding objects. While not being part of a special search tool they provide facilities to represent object names and other features in different ways that the users can choose between. Thus, individual preferences e.g. in sort orders and naming are supported. The tree-like hierarchical presentation as well as the possibility to determine the sort order are very helpful for location-based finding which widely used when working with user interfaces based on the desktop metaphor (cf. Barreau & Nardi 1995, Fertig et al. 1996). In the same workshop a search tool modified from the original search tool was introduced. This prototype 0 contained all the functionality of the original search tool except that a person could only perform a search on her own POLITEAM desk which on one side meant that someone searching could not violate someone else's privacy because she could simply not access other electronic desks, but that on the other side cooperation and team work which POLITEAM focuses on were extremely hampered. Moreover, the response time for the search results became very long since restriction to the desk of the initiator of the search made it necessary to first search all objects on her desk which included a time consuming check for every object in the system and then in a second search restrict the objects on the desk to those for which the search criteria applied.

Prototyping

After the interviews and workshop I and workshop II we felt we knew enough to program a prototypical search tool that was to incorporate what we had learned from the interviews and workshops. Two alternatives were to be considered. The first was to change the original search tool to fit the new requirements. While this would have had performance advantages the means of changing the original search tool provided by LinkWorks™ were not powerful enough to have us implement the features wanted. So in trading off performance for flexibility we decided to use an external programming language for the search tool prototype and access the LinkWorks™ objects by means of the LinkWorks™ programming interface.

We choose Visual Basic as programming language and created a search tool that met the requirements in different ways. It included possibilities to search according to different criteria, among them the name and class of an object, the name of the person who created or owned or changed it, and the date or period when it was created or changed. It was also possible to search for a key word or search the complete object (usually a text object) for a text string. Moreover, to support the communicative aspect of the search, a button to activate the e-mail component of LinkWorks™ from the search dialog was implemented.

A major improvement was the distinction of the area where an object was found. For every object found it was indicated whether it was found on the searcher's own desk, on someone else's desk or in the archive of the group. Knowing this the most interesting objects could be picked. For them a link was created in the search result window of POLITEAM.

The indication for the found objects where they were found is a first step towards a conflict management necessary for a search tool for groupware and groupware in general (cf. Wulf 1995 for a general treatment of conflicts in groupware). Such a conflict management could then handle how objects are treated depending on where they were found, e. g. if the person on whose desk the object was found works in same project as the searching person and the like.

Figure 2: Input dialog of search tool prototype.

Figure 3: Output dialog of search tool showing where objects were found.

Developer Workshop and User Evaluation Workshop

This prototype was presented in workshop III with developers and project members working on the training and support of users. They suggested some minor changes concerning the handling and proposed to incorporate

the possibility to open a video channel for communication about the search from the search tool dialog as soon as video is available for POLITEAM.

The changes were made and the resulting prototype was presented to three users from one of our application partner organizations in workshop IV held at the University of Bonn. Its primary goal was the evaluation of the functionality and user interface of the new search tool. Two of the three users had been interviewed in the initial phase of the redevelopment. By this time the three had used POLITEAM intensively for about 10 weeks. We did not just want to give a demonstration of the search tool but provide a chance for hand-on testing. In order to support material for a discussion of the roles of a searcher and a person “being searched on” we prepared five search scenarios. This was done by rebuilding parts of the structure of the desk the users knew from their daily work and providing computers in two separate rooms to represent two users of POLITEAM. We planned to have them search the system including other people’s desks for a file they needed to proceed with their work and find out what would happen on either of the both computers. Some of the aspects that were meant to be raised by the scenarios were already discussed when we talked about the functionality of the search tool since the three users were experienced and interested enough to recognize what chances and problems might come along with the search tool. They even started a discussion of the different roles of a searcher and a person being searched on by themselves. Thus, it proved to be an advantage that they already had experience with POLITEAM so they could well imagine the search tool in their daily work. For example a user imagined his boss working on the computer late at night searching for documents containing certain key words and stressed the importance to be able to create private domains that others could not access with the search tool. After using the search tool for some of the scenario searches we had prepared and some searches initiated by themselves the users made concrete suggestions on how to improve the input dialog in stating that they usually did not know what a certain person had to do with an object, i.e. if she was the owner, creator or had changed the object, only that she had some relation to it. So they suggested that in the search dialog section where the creator, owner and changer of a document could be specified there should not be three entries but just one so that a person could be specified as having to do with an object with the option to say if she was the owner, creator or changer if you knew. Thus, the former need to put in three times the same name for creator,

owner and changer and connect them with a logical OR is reduced to just pick one name.

State of Work

After prototype 2 has been discussed in workshop IV the changes to it suggested by the group of POLITEAM users will be made so that the resulting software will be ready for release with the next POLITEAM version. With the search tool introduced then the three main problems that arose with the original search tool (unintended deletion of files, user interface, conflict potential) will be solved or prepared to be solved after a process of participatory and evolutionary software development. Moreover, by new ways of representing objects in a hierarchical tree-like structure and with the chance to order objects by different criteria the refinding of objects on user's own desk is considerably improved.

First important steps for the system's conflict handling are made. The new search tool incorporates some prerequisites of conflict detection in showing where the objects were found before they are picked for the search window. The conflict potentials caused by the activation of POLITEAM's group-related functionality require a special module for conflict management for POLITEAM which can then be used by the search tool and which will provide ways to detect and solve conflicts e.g. by informing someone that their desk is searched or giving them the chance to veto against it.

Discussion

The course of the development of a search tool for POLITEAM has shown that an evolutionary and participatory approach for the development of groupware is promising. The different participatory techniques used brought different insights:

- Feedback from the POLITEAM users to the user advocates showed aspects of their cooperative work practice.
- Interviews helped to understand how people search at their workplace and what the requirements for a search tool from the viewpoint of persons searching and 'being searched on' might look like.

- Workshops with POLITEAM users brought up group-related aspects of system use and increased the users' and developers' understanding of conflicts raised by system use.
- A special workshop to present the search tool prototype to POLITEAM users and have them evaluate it in a first step allowed for fine-tuning the search tool to the needs of the application partner and hands-on experience helped to deepen the users' understanding of the conflict potential on a more concrete level. Here, we particularly profited from the fact that the three users were very interested and above-averagely competent in working with POLITEAM.

However, our activities would have benefited from a workshop particularly focusing on the potential conflicts of searching on other persons' electronic desks and discussing the implications with a group of POLITEAM users at an early stage of the development process. This could have helped the developers to learn about the handling of this issue in a concrete organization and give hints for the implementation while users could have become more aware of the implicit rules of their organization and the technical potential to reveal and support them. Unfortunately, the limited amount of time on the application partners' side and the resources provided for the development of the search tool as only one of many of the POLITEAM activities did not allow for such a workshop.

Moreover, the decision to develop a new search tool rather than improving the existing search tool mainly depended on technical considerations. While LinkWorks™ provides some mechanisms to modify or enhance the system's functionality these mechanisms are still not flexible enough since they impose restrictions to the desired implementation.

The STEPS approach (see figure 1) taken as a basis proved to be helpful as a rough guideline for development. Unlike described in the STEPS model and unlike most of the activities within POLITEAM the redevelopment of the search tool was not preceded by the usage of the respective functionality of LinkWorks™ since this was considered to cause too many problems for the application partners. The development activities described above can be located in the production phase of the STEPS model. If we had decided to change the original search tool to fit the new requirements rather than redeveloping it the activities might have been considered to have more of an *adaption* than of a *production*. In that case the development activities could

have been located in the application phase of the STEPS model enhanced by the common activity of adaption as suggested by Wulf & Rohde (1995).

Conclusion

Two more general aspects of introducing the search tool within the POLITEAM framework deserve attention. One of them is the usage of participatory design methods and techniques for the introduction of groupware functionality. Our case study supports the notion that both group workshops and having end users take roles as activator of and someone being affected by a groupware function help to create cooperative awareness. Thus, the concept of perspectivity originally meant to bridge the gap between users and developers is enhanced to let end users get an impression of how other groupware users are affected by their use of functions. At the same time this helps to understand the actual work practice and to make explicit who may cooperate with whom in which way.

The second aspect that our case study contributes to is the introduction of a *generic* groupware product into an organization. Considering the growing need for technology for cooperation and communication inevitably most of the groupware applications installed and used in the future will be generic applications that are adapted to the needs of a special organization. The usability and success of this groupware will to a large degree depend on the quality of this adaption.

Generally the disadvantage of a commercial off-the-shelf product is that it ignores specific social and organizational concerns and users are not known at the time of initial development (Grudin 1991). By providing both organizational means to introduce the groupware and technical mechanisms that allow for different levels of tailoring (cf. e.g. Henderson & Kyng) this disadvantage might be overcome. On the technical side our experiences lead to the conclusion that the approach taken by LinkWorks™ which is based on object-orientation and provides an application programming interface as well as means to change internal methods looks promising. Still, the mechanisms of LinkWorks™ were not flexible enough to fulfill all our needs. Object orientation also plays an important role in more detailed and concrete suggestions made concerning technical means to support tailorability (cf. e.g. Fischer & Girgensohn 1990, Malone et al. 1992, Mørch 1995). The need for flexible solutions also includes the demand to allow for

unanticipated use by supporting the notion of the medial character of the groupware and avoiding the implementation of rigid user “representations” (Bentley & Dourish 1993).

Probably most important for the introduction of generic groupware is an adequate organizational treatment. Previous work on the area of introducing generic groupware into an organization has shown the need for explicit organizational embedment in order to use the full range of groupware advantages (Orlikowski 1992) and drawn the attention to the interplay of intended and emergent induced organizational changes by groupware use (Orlikowski 1995) that demand technical flexibility. These organizational changes will be analyzed carefully in the POLITEAM project to learn more about the impact of introducing a groupware and have the introduction process benefit from this knowledge.

In many ways introducing a generic groupware resembles the design process for the development of a custom-made groupware. Here, methods of participatory design can be used for participatory tailoring. Research having taken into account the influence of a group structure for participatory design and development can give important hints for methods of participatorily introducing and tailoring generic groupware. For example, Kjær & Madsen (1994) suggest a participatory analysis of flexibility based on a “blueprint mapping” technique to get an overview of the daily work and on an “organizational game” to analyze the need and potential for organizational flexibility. Another closer look at organizational aspects of tailoring that can go beyond the phase of initial implementation is taken in some papers dealing with the sharing of customization files (Mackay 1990; Nardi 1993 Chapter 6; Trigg & Bødker 1994). While these findings are not explicitly related to groupware they involve group activity to customize software used by a group. The papers stress the importance of local experts who know the work practice well enough to provide adequate customization.

Still, more work has to be done on the impact of group particularities on the use of groupware functionality, how roles are represented in groupware and how conflicts can be detected and mediated that are induced or made visible by system use. Here, many questions remain open (cf. Kahler 1995). How can we proceed when introducing one groupware for different organizations? How much tailoring can and must be done? What can participation not only in the process of design but also in the process of introducing a system look like? How can we share responsibilities for

customizing groupware for an organization between users and developers? How can we support participation for system introduction and customization by preconfigured systems?

This paper has provided a case study indicating that continuous work with an evolutionary and participatory approach to the development of groupware and its introduction may help to answer these questions.

Acknowledgements

I would like to thank Guido Krüdenscheidt who implemented the search tool, our partners from the interviews and workshops for contributing their time and patience, and the three anonymous PDC reviewers for helpful comments on an earlier version of this paper.

References

- Barreau, Deborah; Nardi, Bonnie A. (1995): *Finding and Reminding: File organization from the Desktop*. In: SIGCHI Bulletin July 1995, Vol. 27 (3). pp. 39-43.
- Bentley, Richard; Dourish, Paul (1995): *Medium vs. Mechanism: Supporting Collaboration Through Customization*. In: Proceedings of ECSCW '95. Kluwer. pp. 133-148.
- Boehm, Barry W. (1976): *Software Engineering*. In: IEEE Transactions on Computers, Vol. C-25 (12). pp. 1216-1241.
- Boehm, Barry W. (1988): *A Spiral Model of Software Development and Enhancement*. In: IEEE Computer, Vol. 1988 (5). pp. 61-72.
- Ehn, Pelle (1993): *Scandinavian Design: On Participation and Skill*. In: Schuler, Doug; Namioka, Aki: *Participatory Design - Principles and Practices*. Hillsdale, NJ, Lawrence Erlbaum Associates. pp. 41-77.
- Fertig, Scott; Freeman, Eric; Gelernter, David (1996): *"Finding and Reminding" Reconsidered*. In: SIGCHI Bulletin Jan 1996, Vol. 28 (1). pp. 66-69.
- Fischer, Gerhard; Girgensohn, Andreas (1990): *End-User Modifiability in Design Environments*. In: Proceedings of CHI '90. pp. 183-191.

- Floyd, Christiane (1993): *STEPS - A Methodical Approach to PD*. In: Communications of the ACM, Vol. 36 (4 - June 1993). pp. 83.
- Floyd, Christiane; Mehl, Wolf-Michael; Reisin, Fanny-Michaela; Schmidt, Gerhard; Wolf, Gregor (1989): *Out of Scandinavia: Alternative Approaches to Software Design and System Development*. In: Human-Computer Interaction, Vol. 4. pp. 253-350.
- Floyd, Christiane; Reisin, Fanny-Michaela; Schmidt, Gerhard (1989): *STEPS to Software Development with Users*. In Ghezzi, Carlo; McDermid, John A.: ESEC'89 - 2nd European Software Engineering Conference, University of Warwick, Coventry. Heidelberg, Springer. pp. 48-64.
- Greenbaum, Joan; Kyng, Morten (1991): *Design at Work - Cooperative Design of Computer Systems*. Hillsdale, NJ, Lawrence Erlbaum Associates.
- Grønbaek, Kaj; Kyng, Morten; Mogensen, Preben (1995): *Cooperative Experimental System Development - cooperative techniques beyond initial design and analysis*. In: Computers in Context: Joining Forces in Design. Aarhus. pp. 20-29.
- Grudin, Jonathan (1991): *Interactive Systems: Bridging the Gaps Between Developers and Users*. In: IEEE Computer, Vol. April 1991. pp. 59-69.
- Henderson, Austin; Kyng, Morten (1991): *There's No Place Like Home: Continuing Design in Use*. In Greenbaum, Joan; Kyng, Morten: Design at Work - Cooperative Design of Computer Systems. Hillsdale, NJ, Lawrence Erlbaum Associates. pp. 219-240.
- Henderson-Sellers, Brian; Edwards, Julian M (1990): *The Object-Oriented Systems Life Cycle*. In: Communications of the ACM, Vol. 33 (9). pp. 143-159.
- Hesse, Wolfgang; Weltz, Friedrich (1994): *Projektmanagement für evolutionäre Software-Entwicklung*. In: Information Management, Vol. 1994 (3). pp. 20-32.
- Kahler, Helge (1995): *From Taylorism to Tailorability*. In: Proceedings of HCI '95, Vol. 20B. Elsevier, Amsterdam. pp. 995-1000.

- Kjær, Arne; Madsen, Kim Halskov (1994): *Participatory Analysis of Flexibility: Some Experiences*. In: Proceedings of Participatory Design Conference '94. pp. 21-31.
- Klöckner, Konrad; Mambrey, Peter; Sohlenkamp, Markus; Prinz, Wolfgang; Fuchs, Ludwin; Kolvenbach, Sabine; Pankoke-Babatz, Uta; Syri, Anja (1995): *POLITeam: Bridging the Gap between Bonn and Berlin for and with the Users*. In: Proceedings of ECSCW '95. pp. 17-31.
- Krüdenscheidt, Guido (1996): *Partizipative Entwicklung eines Suchtools für Groupware (English: Participative Development of a Search Tool for Groupware)*. Department of Computer Science III, University of Bonn. Diploma Thesis.
- Lehman, Manny; Belady, Laszlo A. (1985): *Program Evolution: Processes of Software Change*. London, Academic Press.
- Mackay, Wendy E. (1990): *Patterns of Sharing Customizable Software*. In: Proceedings of CSCW '90. pp. 209-221.
- Malone, Thomas W.; Lai, Kum-Yew; Fry, Christopher (1992): *Experiments with Oval: A Radically Tailorable Tool for Cooperative Work*. In: Proceedings of CSCW '92. pp. 289-297.
- Mambrey, Peter; Mark, Gloria; Pankoke-Babatz, Uta (1996): *Integrating user advocacy into participatory design: The designers' perspective*. In: Proceedings of PDC '96. Computer Professionals for Social Responsibility. pp. 251-259.
- Mørch, Anders (1995): *Three Levels of End-user Tailoring: Customization, Integration, and Extension*. In: Computers in Context: Joining Forces in Design. Aarhus. pp. 157-166.
- Nardi, Bonnie M. (1993): *A Small Matter of Programming*. Cambridge, Massachusetts, MIT Press.
- Orlikowski, Wanda (1992): *LEARNING FROM NOTES: Organizational Issues in Groupware Implementation*. In: Proceedings of CSCW '92. pp. 362-369.
- Orlikowski, Wanda (1996): *Evolving with Notes: Organizational Change around Groupware Technology*. In Ciborra, Claudio U.: Groupware & Teamwork - Invisible Aid or Technical Hindrance, Wiley. pp. 23-60.

- Trigg, Randall; Bødker, Susanne (1994): *From Implementation to Design: Tailoring and the Emergence of Systematization in CSCW*. In: Proceedings of CSCW '94. pp. 45-54.
- Wulf, Volker (1995): *Mechanisms for Conflict Management in Groupware*. In: Proceedings of International Conference on Human Computer Interaction (HCI) '95. pp. 379-385.
- Wulf, Volker; Hartmann, Anja (1994): *The Ambivalence of Network's Visibility in an Organizational Context*. In Clement, Andrew; Kolm, Paul; Wagner, Ina: *NetWorking: Connecting Workers In and Between Organizations*. Amsterdam North Holland. pp. 143-152.
- Wulf, Volker; Rohde, Markus (1995): *Towards an Integrated Organization and Technology Development*. In: Proceedings of Symposium on Designing Interactive Systems. Processes, Practices, Methods & Techniques. ACM. pp. 55-65.

Sixth Paper

Tailoring by Integration of Components: The Case of a Document Search Tool

Abstract

In this paper we describe the evolutionary design and implementation of a search tool for files in shared workspaces used within an off-the-shelf groupware product. The design is based on the assumption that a useful generic search tool must be highly tailorable. We achieve tailorability by applying an innovative software architecture which allows to assemble components during runtime. In order to understand how people search in shared workspaces and to support the design we employed interviews and workshops with users as well as a field test to understand users' needs. During the design process a series of prototypes was developed by us which were then evaluated by office workers. Consequently, the process described and the lessons learned extend from searching in files as a case via tailorability of software as an answer to the resulting requirements to component architecture as a way to implement this tailorability. The results derived from the treatment of these interrelated aspects constitute the core and value of this paper.

Introduction

The POLITEAM pilot project dealt with the groupware support of distributed governance as a reaction to the relocation of parts of the German government from Bonn to Berlin. Among a number of research issues was the search of files in a shared workspace of a groupware system that the POLITEAM project introduced at several sites of the German public administration (see section POLITEAM: *The Context of the Search Problem*). While the issue of searching for files was the starting point of our research, we considered tailorability of software to be one of the possible solutions on

a theoretical level for the multiple and changing user requirements for a software to search files within a group. On the practical level, using software components appeared to be a feasible option for implementing this tailorability. Thus, this paper is an account of our work on a component-based tailorable search tool and describes the lessons learned. While one might argue that this fails to focus on a single key issue we think that our case benefits from the intertwinement of these three levels. They deserve and require the joint treatment in the paper that we gave them in the process. Our research was guided by two interrelated questions: The overall question *How can we design for searching in groupware?* is concerned with software-technical as well as procedural aspects and issues of the user interface and the underlying functions. The more specific question *How well are components suited for runtime tailorability?* relates to a particular solution for designing for searching in groupware. The given answer does not only contribute to the overall question but constitutes a value of its own for a range of applications.

The goal of this paper is twofold. Firstly, the paper is *descriptive*: We describe the lengthy process of different steps in developing a search tool used in a groupware setting. Secondly, the paper is *prescriptive* in that it reflects on the process and depicts the lessons learned. By following these two lines we aim at providing a deeper understanding of what we did, of what relevance it has for searching, tailorability and component architectures and their relation and we aim at providing support for others facing similar design challenges.

The development of the search tool in several steps is accompanied by the participation of users and accomplished in an evolutionary way by implementing several consecutive prototypes of search tools each based on empirical and theoretical insights. The empirical insights are derived from the evaluation of the previous version of the search tool or empirical work conducted before the introduction of the current version. The theoretical insights come from reading related literature where the relation sometimes only became clear during the process.

The structure of the paper reflects the process. After an account of the state of the art in searching, tailoring, and component architectures, the process of the development of three search tools is described in chronological order including methodical issues like the form of user involvement chosen,

technical issues and empirical results. The paper is concluded with a discussion of what we learned and answers to our research questions.

State of the Art

In this section the state of the art of searching for files, tailoring software and component-based software architectures is provided.

Searching for Files

Searching in computer systems is one of the most fundamental tasks that has to be performed while working with a computer. In a distributed system, which is used for collaborative work, searching for files as well as searching within the files is crucial for the system's success, mainly because good search results are vital to handle the huge data amount of such systems. It is also important for the discussion on collaborative work to keep in mind that the success of any searching activity is highly dependent on the collaborative working experience of the people performing the search and of the respective software settings, particularly access control.

Several empirical studies deal with file organization in the electronic office (cf. Malone 1983; Suchman & Wynn 1984; Barreau & Nardi 1995; Rao et al. 1994). These studies observe the organization of electronic documents from a single user and static perspective. In recent years, the production of documents became a collaborative activity supported by networked computers for many people (cf. Wulf 1997). This constitutes a particular need for research on aspects of the organization and search of files in shared workspaces.

Tailoring Software

Our early experiences with the search tool in the POLiTEAM setting showed a wide range of requirements from different persons in different work settings using the system. Meeting different and even mutually exclusive requirements can be achieved by making the used software tailorable. Tailorability is a software attribute which allows to change certain aspects of the software to meet different user needs.

Tailorability is a desirable software feature for several reasons:

- The *diversity* of requirements posted by different individuals and organizations must be taken into account when implementing, buying, or using a generic software that is supposed to fit different settings. What may be appropriate for a Federal Ministry with hundreds of employees might not work at all for a shoe-selling small enterprise using a software system.
- The *uncertainty* about the exact work practices and procedures even in the perception of the workers makes it necessary to leave room for alternative ways of performing tasks (Trigg 1992). In many cases there is not only one way to perform a certain task, each user has a different way of working with a software.
- The *dynamism* of individual and organizational work requires software to change over time. The structures of work organization and collaboration may vary considerably in relatively short time periods.

Tailoring software encompasses different dimensions, e.g. the initiator, actor, object, aim, time, and scope that should be considered already at the time of system design (cf. Stiemerling et al. 1997). It is widely agreed that tailorability is one of the major future challenges in the design of interactive systems (e.g. MacLean et al. 1990, Bentley & Dourish 1995, JCSCW 2000). Henderson & Kyng (1991) consider tailoring to be an activity that continues design in use. All of the contributions stress that the discussion about tailoring should not only be lead in terms of technical measures, but that tailoring software is an activity that is deeply rooted in personal habits and preferences as well as socio-organizational circumstances and dynamics.

A closer look at organizational aspects of tailoring is taken in some papers dealing with the sharing of tailored files (Mackay 1990; Nardi 1993). Trigg & Bødker (1994) point to the fact that the possibility to exchange tailored artifacts can have a standardizing effect. While these findings are not explicitly related to groupware they involve group activity to tailor software used by a group. The papers stress the importance of local experts who know the work practice well enough to provide adequate tailoring. Carter & Henderson (1990) claim the necessity for a „tailoring culture” within an organization because tailoring not only poses technical problems but should be considered as a relationship rather than a property. MacLean et al. (1990) go beyond observing by introducing the „Buttons” system into part of an organization by means of which tailored objects can be sent around and be modified. The above-mentioned diversity of requirements

poses a particular challenge for tailoring software to a group's needs: on one hand the diverse requirements of the different members of the group must be somehow combined or moderated into a convention or standard for group usage; on the other hand it should still be possible for a group member to tailor within the limits of the convention.

Component Architectures

To put tailorability into practice one has to develop software architectures that allow for runtime flexibility to avoid a breakdown in the flow of work. For some time now, component-based architectures are available for software developers (e.g. with the `JAVABEANS` model). Our research focuses on the question whether specific component-based architectures can enable end-users to modify existing software to fit their needs or create new functions by assembling existing pieces. In this way component-based architectures should become a means to alleviate the climbing of the steep „tailorability mountain” (MacLean et al. 1990).

Allowing the construction of software from components gives the person who puts the components together a flexibility in construction that encompasses the usual ways to tailor software by far without being as difficult as programming. Therefore it is reasonable to give this form of flexibility to end-users or local experts to compose software or change existing compositions to match their needs.

Component oriented programming in general (see Szyperski & Pfister 1996) is motivated by the successes classical engineering disciplines like electronics or mechanical engineering have had with building complex artifacts from standardized components (e.g. transistors, resistors, cogs, or screws). Taking into account this motivation, a software component can be defined as *„a unit of composition with contractually specified interfaces and explicit context dependencies only. Components can be deployed independently and are subject to composition by third parties.”* (Szyperski & Pfister 1996, p. 130).

Components have been successfully employed to support the design of graphical user interfaces. Application builders like `LOTUS BEANMACHINE` (component model: `JAVABEANS`, see JavaSoft 1997) often provide generic visual design elements (e.g. buttons, text-boxes, combo-boxes), which are configured and composed during the design process to yield domain-

specific applications. The notion of components, however, has been applied to areas of software engineering other than GUI-design, as well.

If the architecture consists of multiple layers of nested components (hierarchical component architecture), tailoring operations are possible at several different levels of abstraction and complexity. Components on the higher levels of the hierarchy could be closer in semantics to the application domain (e.g. the bookkeeping component of a business software package), while components further down the hierarchy could be more technically oriented (e.g. the TCP/IP-protocol component). Thus, a hierarchical component architecture could provide appropriate levels of tailoring for both a bookkeeper and a system administrator.

In our work (Stiemerling 1997, Stiemerling & Cremers 1998, Won 1998) we are investigating the use of components for tailorability of complex and distributed applications after initial development. Component architectures are quite attractive for tailoring, because they support a number of different tailoring interfaces, from simple parameterization (Henderson & Kyng 1991), over visual programming (Nardi 1993) to programming by modification of examples (Nardi 1993, Mørch 1997). Particularly runtime composition by users or local experts can support tailoring activities because its effects are immediately visible and there is no need to engage into activities like compiling - generally performed by experienced system administrators only - or rebooting a machine which is often considered to be a breakdown of the flow of work.

The advantages, which component oriented software engineering is hoping to provide for developers can be harnessed for users' tailoring activities:

- facilitation of re-use by third parties as facilitation of re-use by other tailors;
- speed-up of development processes and reduction of development costs as avoiding to reinvent already existing tailoring artifacts provided an infrastructure to exchange them, and
- higher quality because standard components are less prone to exhibit errors when they have already been used and tested in prior projects or by other users and local experts, respectively.

POLiTEAM: The Context of the Search Problem

The POLiTEAM project was a collaborative software development project in which the target organizations required technical support for distributed collaboration. The main function of the POLiTEAM system was to supplement paper work processes with electronic work processes in a government ministry. To accomplish this, POLiTEAM offered a shared workspace, electronic circulation folders and e-mail functionality (cf. Prinz & Kolvenbach 1996). An already existing groupware system (LINKWORKS by DEC) was chosen and adapted to specific user and situation requirements (see Klöckner et al. 1995). A collaborative and evolutionary approach was used in the design, allowing modifications to be made over time. These modifications were reported as beneficial by designers and users (cf. Wulf 1997). Within this design process user advocates played a special role. These project members visited the sites regularly, provided support to the users and thus were able to perceive user requirements in a direct way (cf. Mambrey et al. 1996). The project started in May 1994 and ended in December 1998; since January, 1995 the system had been installed. While the search tool had been a design issue since the very beginning of the project, the component-based version was developed in the last year of the project.

Obtaining the user requirements was accomplished with the help of users at two different government organizations: a German Federal Ministry (referred to as FM in the paper), and the State Representative Office of a German state (referred to as SR in the paper), both located in Bonn. In the FM, in the department where the system was installed, we found varied employee roles: one unit leader, six ministry employees (responsible for specific content areas of the ministry), and three typists in their own service unit. In the SR Body there were about 30 people working who represented the interests of their state in the federal government's legislation process. The organizational structure of the body mainly consisted of sections which represented state ministries. Most of the sections just consisted of the section manager. Before the introduction of POLiTEAM, these sections were supported by three typists, who worked at the typing pool. Like in the FM, the employees collaborated using the shared workspace and e-mail.

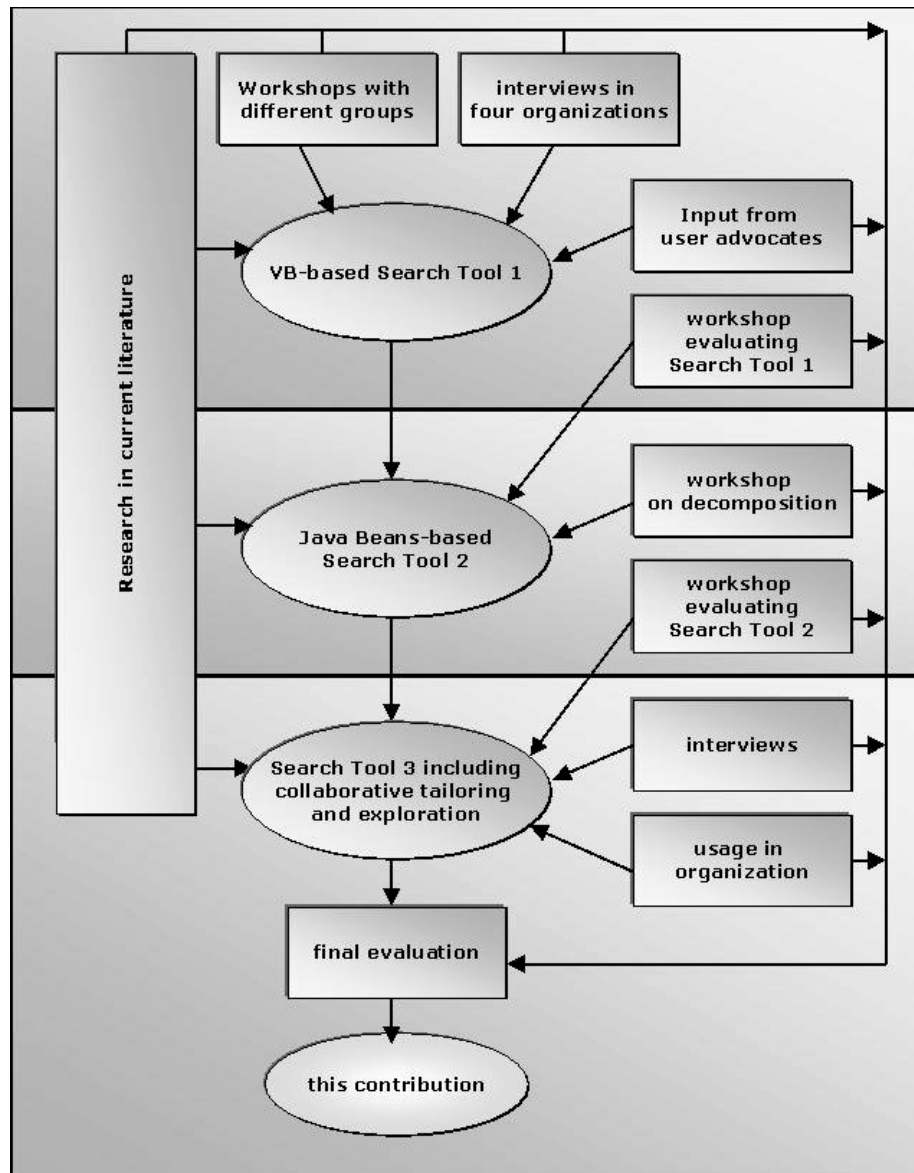


Figure 1: Steps in the process of designing several versions of search tools

Among the multitude of challenges that POLITEAM had to face dealing with such a complex issue as the introduction of groupware in large administrations, searching of files was of particular interest. This is due to the fact that the administrative work is centered around text documents that are often worked on by different people. While this can sometimes be

handled by sequential workflow mechanisms, the collaboration often is too unstructured for this rigid kind of support. Therefore the possibility for electronic search offers a benefit particularly for the less structured collaborative parts of administrative work. Besides, searching files electronically within and possibly even beyond common workspaces raises many questions of access rights and the ambiguity of visibility and privacy. To meet the resulting requirements, in the course of the POLITEAM project we developed a tailorable search tool for the participating users to search for files in the shared workspace. For the development of the search tool we proceeded in three major steps. Taking a participatory and evolutionary approach we first developed Search Tool 1 using Microsoft Visual Basic (VB) as a means for prototyping to meet general needs with a default setting for a generic search tool (cf. Kahler 1996). In a second step we used the evaluation of Search Tool 1 for a reimplementaion in Java. The resulting Search Tool 2 allowed end-users and organizations to tailor the tool to their particular search requirements (cf. Won 1998). Finally, Search Tool 3 (cf. Engelskirchen 2000) provided mechanisms to support collaborative tailoring activities and the exploration of the tailoring functionality. Figure 1 gives an overview on the design process.

Search Tool 1: Involving Users

The basic version of LINKWORKS included a tool that allowed the user to search for any object independent of its actual location within the system. Discussions with users revealed that this search tool was not well enough designed to be used by our two government organizations involved, since the issues of privacy and unintentional manipulation of shared files were not satisfactorily dealt with. Also, possible conflicts about snooping around on others' desks were not considered. So we decided not to use the original search tool in our partner organizations but to redesign it.

Requirements from the Field

To identify the basic functionality of a search tool in groupware and possible enhancements we conducted 10 interviews with interview partners from four different organizations lasting about 30-45 minutes each. Besides the interviews, in this first step of the redevelopment of the search tool, workshops were held with a group of users in the FM where searching was

discussed. Several aspects of a group-related tool for electronic search in a shared workspace proved to be of importance.

Privacy Issues

On one hand users want to have full access to all the documents stored in the system on the other hand they demanded that „their” documents should not be accessible for other users. In the role of a person searching actively, the interviewees pleaded for a nearly unlimited access to electronic search arguing that this would be helpful and necessary for collaboration and adequate for team work. They were much more privacy alert when they took the role of a person affected by someone else’s electronic search. Here, many of the interviewees said they felt uncomfortable knowing that everyone could have a look at their folders and considered this as an unwanted intrusion. This „I want to see yours but you may not see mine” attitude obviously requires technical and social mechanisms for mediation.

Beyond these individual preferences the informal ways of collaboration are specific for an organization. In the FM searching on other people’s desk was a taboo except for the head of department and one other person who was responsible for the registry. The head of the department of the FM mentioned that considering the different organizational standards about standardization there is a need to be able to set the borders for tailorability for the search tool so that the tailoring of the tool would not go beyond reasonable limits. This, however, is the key point for tailorability: what *reasonable* means cannot be foreseen when specifying a generic software particularly for work in groups.

User Interface

The users had different preferences about what to do with found documents (e. g. create a copy or create a link). Furthermore they did not agree in which way the search result should be ordered (e. g. by name or by date). This again showed the need for individual and organizational tailorability of the search tool.

Implementation

We employed Microsoft Visual Basic in order to be able to quickly develop the first version of the new search tool.

The input dialog of Search Tool 1 considers the most frequent requirements from the interviews and workshops. There are several input fields which allow the user to specify the search request. The output dialog of Search Tool 1 takes privacy aspects into account by dividing the output into files found on one's own desk, someone else's desk or in the registry. A link from the found items can be created in one's own work space.

This first prototype did not incorporate all the demands of the interviewees and the participants of the workshops for two reasons. First, we wanted to force the privacy discussion, so we only split the search result into three groups. Second, we just allowed to create links on the found objects. The first prototype of the search tool was presented only a short time after the introduction of LINKWORKS and most of the users had no experience with groupware systems or even computer systems at all until then.

Evaluation

Search Tool 1 was now presented to ten project members (users and user advocates) during a workshop. The discussions verified existing requirements and helped to identify new demands. Besides, our user advocates who supported the pilot users during their daily work added some requirements to a new version of the search tool.

User Interface

Most of the users required a spatial separation of the search request panel and the displaying components within one window so they could see their search request while viewing their search results. In Search Tool 1 we used two windows, one that allowed the input of several search criteria and another that displayed the results of the search. Therefore the new search tool should have at least two components: an input component and a display component which have to be displayed in the same window.

The users had no common idea about the representation of the result. Some users wanted to have exactly one window where all the objects found in the system are shown. Others would have liked the search result to be divided into several subsets. The found objects then should be sorted by certain criteria. For instance some users wanted to see the documents in two separated windows: one for documents which lay on their own virtual

desktop and another for the rest. Others could imagine that the documents are ordered by their name or by their date of creation.

Search Criteria

All users had different preferences how to search for an object. For instance, some users searched for a document by its name, others by the date of the last change or they searched for the document's owner. Therefore no simple way to decide which of the possible search criteria should be asked for by our search tool could be determined.

Tailorability is one way to meet these different requirements. So we decided to implement a new version of our search tool which was supposed to offer several aspects of tailorability. Therefore, Search Tool 2 was implemented by using component-based architectures.

Search Tool 2: Towards Component-based Tailorability

In this section we describe how we used a component architecture to create a tailorable search tool. The idea was to use the established work on components and wiring diagrams to connect these to enable users and local experts to create, modify and test search tools during runtime.

As described in the section *State of the Art* component architectures can be used for the design of applications. In this section we will focus on the changes to an application during its use. Taking a closer look at the „component model” used in Visual Basic for Search Tool 1 one can easily see that after the compilation of the application the components are not relevant anymore. Components can only be bound together in a design environment and therefore facilitate the development of software. But after compilation the result is a monolithic software that can not be decomposed. In contrast to this component model we will in the following concentrate on applications that consist of components that are dynamically bound. If these applications have to be tailored one can simply add or remove one component or change the connections between them. In order to realize this runtime tailorability for Search Tool 2 we employed the Sun `JAVABEANS` component model.

In the case of Search Tool 2 a layered architecture (see section *State of the Art*) is used for experienced users to develop their own search tool selecting

and combining up to about twenty components whereas a beginner would take just two nested components which can contain many atomic components and combine them to a simple search tool. This approach is described below in more detail.

Introductory Workshop

From our experiences from Search Tool 1 we already had a good idea about searching in groups. For our aim of providing components for building different search tools we needed a better understanding of what a reasonable deconstruction of a search tool into components would be. This was one of the foci of a full-day workshop (workshop 1 of Search Tool 2) we held with 9 persons from the FM and SR. The analysis of several other search tools had shown that a first approach to a composition would be two distinguished parts related to the chronological sequence of searching for electronic files in a shared workspace. First, a search is performed according to a specific search request, then the results are presented and can be used for further work. Most of the common search engines are divided into two parts in a similar way. The workshop confirmed this and provided further hints particularly for different in- and output switches taking into consideration e. g. age and name of files.

The division into two parts seen from the user's perspective results in the division into three parts that we provided: the latter can be deduced by looking at the three types of components described below. Here the output components are distinguished from the so-called flow components which are used to perform a search and to do some work on the search results like splitting it into two output streams (switch) or getting some more attributes (e. g. date of creation, last change) of them.

Implementation

Search Tool 2 was developed by taking into account the diverse requirements that evolved during the above mentioned workshop. We chose the `JAVABEANS` component model (see Java Soft 1997) that allows for dynamical binding of components as a basis for our implementation. First we will describe the new search tool from a more technical view. Then we will have a more thorough look on the user interface.

The Components

Based on the results of workshop 1 of Search Tool 2 described above we decomposed Search Tool 1 into several different components which are divided in three categories: input components, data flow components, and output components (cf. figure 2 and 3).

The top of figure 2 shows some of the implemented input components. These components are used to specify the search and to trigger an action. For instance, one can enter the name of the objects that are to be found and start a search.

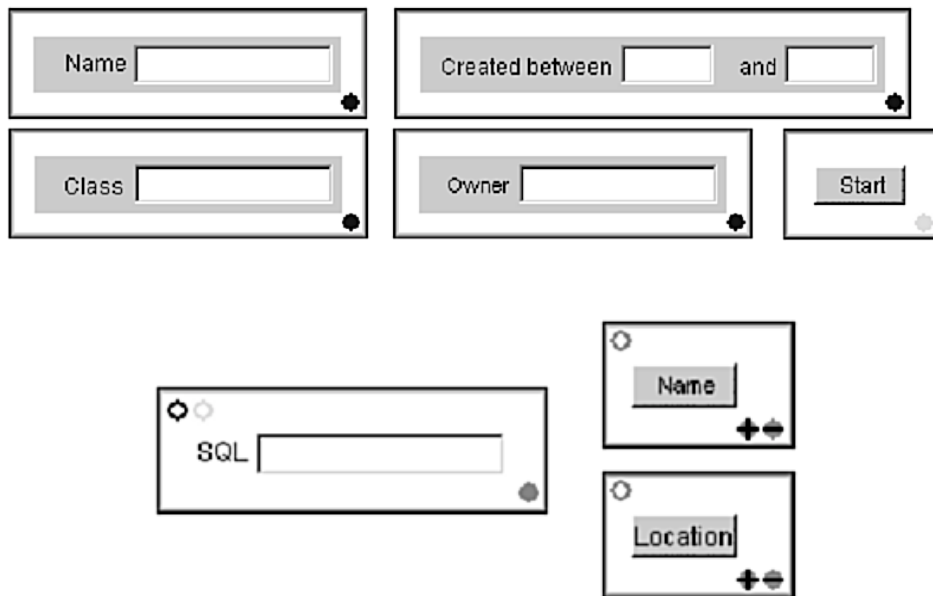


Figure 2: Some input (top 5) and flow (bottom 3) components

The bottom of figure 2 depicts some flow components including the search engine. This component has to deal with the input parameters generated by the input components and has to build up the search results. The search engine connects the search tool with the LINKWORKS database via the application programming interface. It transfers the inquiry of the users to the database and receives a list of retrieved documents. Two other components which can be used to prepare the visualization of the search result are the name switch component and the location switch component. These are

components that are used if the result of a search has to be split respecting a condition that was specified beforehand („+” meaning condition is true). The name switch divides the resulting files by their names (e.g. starting with A-M or starting with N-Z). The location switch divides the resulting files by their location in the system (own desktop vs. other desktops). To get a more refined presentation of the result one could imagine many different switches.

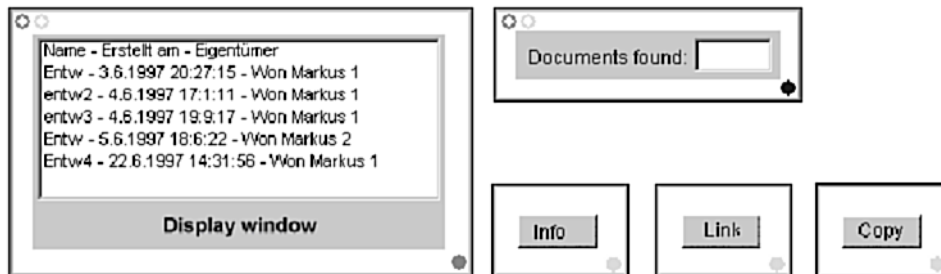


Figure 3: Some output components

Figure 3 shows some output components. The display window shows the files found. The user can select which of the document's attributes are shown and how the retrieved documents are ordered. Other components which can be used to visualize the search result are the counter component („Documents found” – added in Search Tool 3 shows the number of found objects and serves to get some information about files on others' desktops without showing any of their attributes) or the info, link and copy components which allow to show additional information, create a link to the file and to copy a found file. In the following we will describe the composition of a search tool using these components and we will illustrate their interaction. For the sake of simplicity we will now concentrate on six atomic components: start button, name input component, search engine, location switch, result counter and result list (see figure 4).

To connect the components with each other we implemented „wiring-instructions“ which define how the application is to be composed. In order to compose the components the tailoring environment allows for wiring operations, which support connecting two different types of ports. Empty circles indicate input ports, full circles output ports. To support users in wiring the components appropriately, input and output ports, which can be connected, are presented in the same color (in this paper they are shown in

the same gray scale). The semantics of the components are best understood by regarding the simple example in figure 4.

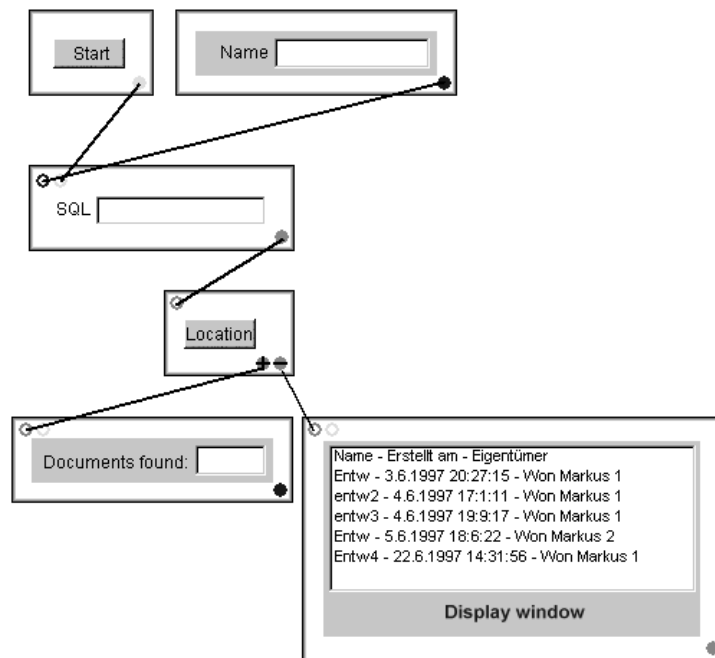


Figure 4: Simple search tool example

The search engine is triggered by the start button. To specify the search query the value is entered in the name input component. The search results are fed into the location switch which is parameterized to channel all documents found on one's own virtual desktop into the right result list, while the number of documents found elsewhere is shown in the counter window on the left (the parameterization of result switches, buttons, and the search engine is not shown here).

A more complex example using the layered architectures is given below (cf. figure 6).

The Runtime and Tailoring Environment

We developed a prototype of an integrated runtime and tailoring environment which serves as basis for the deployment of component based application (see figure 5). These applications are defined by a set of implemented components.

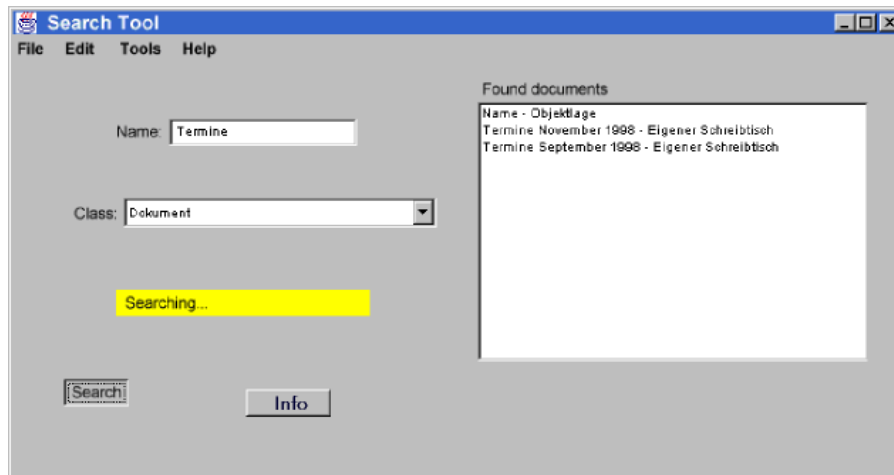


Figure 5: The integrated runtime and tailoring environment in runtime mode

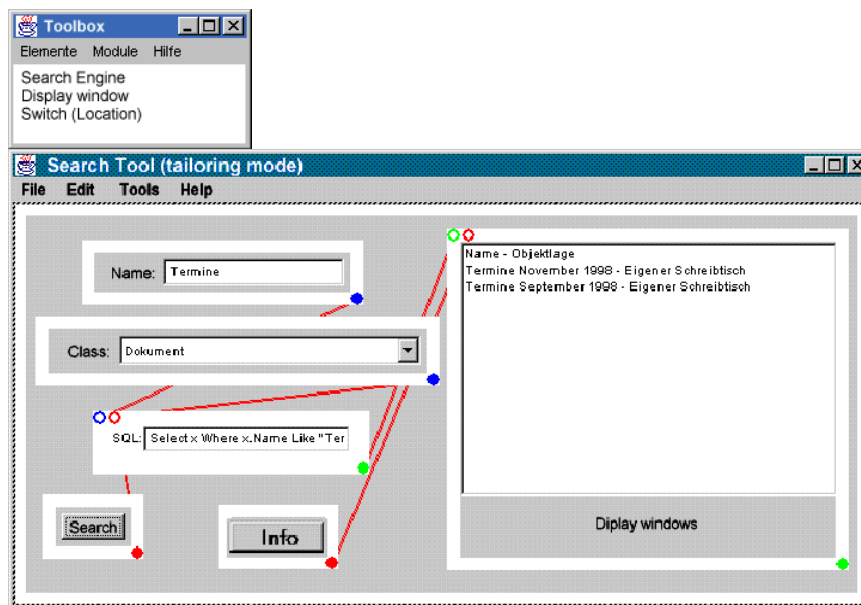


Figure 6: The integrated runtime and tailoring environment in tailoring mode

When the application is started within the environment, the default wiring-instructions are read first. According to these, the necessary components are instantiated and connected. For a more detailed discussion of the tailoring environment see Won (1998). Here we want to abstract from these

technicalities and focus on how the environment is perceived by the end user.

If the application is running on a windows-based operating system on a regular basis (the whole environment is implemented in Java and thus platform independent), nothing unusual should be visible, expect some way to leave the „use-mode“ and enter the „tailoring-mode“. If the user decides that the application needs tailoring, he or she enters the „tailoring mode“. Figure 6 shows the tailoring-mode for one specific instance of the search tool. Here we want to focus on the general concepts of the tailoring environment. The little dots at the edges of the components represent the ports, i. e. the interfaces for interaction with other components. As described above the search tool employs a flow-oriented metaphor (data flow through the application), we have input ports (empty circles) and output ports (full circles). The lines indicate how the components are wired with each other.

The user now can do everything but change the implementation of the components which are delivered in binary form. He or she can delete components, instantiate new ones (by choosing a component from the toolbox menu), change the wiring, or change the hierarchical structure. These operations ensure flexibility of the approach. Whether the flexibility offered is satisfying of course depends on the set of available application components.

Evaluation

To evaluate the design of the component-based search tool and its integrated runtime and tailoring environment, we held another workshop (workshop 2 of Search Tool 2). We were mainly interested if the users were willing and able to handle the tailoring functions of the new environment and if the existing version of the component-based search tool could satisfy the different user requirements.

This workshop was held at the University of Bonn’s research lab. Eleven participants joined the workshop. Four of them were employees of the SR - a section manager, an administrative clerk, a secretary and a clerk who provides local support to other users. Moreover, three user advocates – two working with the SR and one working with the FM – participated in workshop. The other participants were members of the POLITEAM project involved with the design of the search tool. The discussions that occurred

during the workshop were documented by the project members and transcribed. The quotations presented in the paper are taken from this transcription and translated from German by the authors.

In the beginning of workshop 2 of Search Tool 2 one of the user advocates gave a short presentation. Comparing Search Tool 1 with the new one, he gave a survey on the new functionality. To introduce the integrated runtime and tailoring environment the user advocate referred to the „Lego“-metaphor. Then one of the designers gave a presentation on different search tools and the tailoring environment on a computer, where a LINKWORKS client was installed. The search tool active at the beginning of the presentation was a simple tool which just allowed to search on ones own desktop by specifying three search attributes. Activating the tailoring menu, the users could select among three different alternatives of the search tool. These alternatives had been assembled beforehand and were represented in the menu by a term that tried to express their features. From this selection menu it was possible to switch into the tailoring environment where a new search tool could be tailored by connecting the existing components. After the presentation the four users from the SR were asked to apply the tailoring environment and build a new search tool by themselves. All of the four users were willing to experiment with the tailoring environment. Receiving some support by the project members, they were able to construct search tools of different complexities. In the following we will give an overview on the aspects discussed during the workshop.

Understanding Component Architectures

As described above Search Tool 2 consists of several components as well as a runtime and tailoring environment. This approach was clear to all users. Additionally, even the hierarchical constructions were understood without any problems by more experienced users.

Additional Components

Looking at the functionality provided by the components, a couple of further design requirements were wished for by the users during the workshop.

- **Full text search**

Concerning the input attributes for the search engine the users asked to be able to search for arbitrary words inside the documents. Often they could not remember any of the given document attributes but a key word inside the text.

- **More expressive display components**

The users also asked for a new type of display window for objects found. Some required that this component should display the location of a found object.

- **Showing the content of the document**

Having found objects the users required being able to access them directly. In the prototype as well as in the original search tool they can just create a copy of or a link to such a document. Links and copies have to be accessed via another window. This leads to additional efforts in interacting with the system.

- **Switches**

The prototype offered just two switches: one to distinguish between the different desks a document was found on, and one to distinguish between different object names. Appreciating the concept of separate display windows, the users required further switches.

The section manager who has to handle similar issues periodically was asking for a switch which would display the documents which are older than three months in a window separated from a window that shows documents which were written more recently. He also asked for a switch which would allow to display objects in different windows depending on the question in which of his own folders these objects were found. He said that he is often searching for a document referring to just one of the political areas he is responsible for. As he stores such documents in one folder with corresponding subfolders, he just wants to see documents found in that folder. Concerning the electronic registrar one of the administrative clerks suggested to be able to distinguish between the folders of the Bundestag (first chamber of parliament) and the Bundesrat (second chamber of parliament). Moreover, one user suggested that it would be nice to distinguish between those documents, which are on one's own desktop and those which are just accessible via document sharing.

User Interface

After the presentation only two of the four users said that they would be willing to use the tailoring environment to build their own search tools. These users felt that the graphical interface for connecting components is too complicated to handle without being supported. This feeling is expressed by the final statement of the administrative clerk before leaving the lab: „*Please make it simple, we are just users!*“.

- **More expressive descriptions and additional context help**

From the users' tailoring experience during the workshop the following requirements to improve the tailoring environment came up. First of all, the users asked for a better description of the different components they could select from. In the prototype the developers had given the components names which were difficult to understand by the users. Then the users asked for more appropriate names and for icons in the select boxes which were supposed to symbolize the meaning of individual components as well as whole search tool alternatives.

Apart from a more intuitive naming the users asked for a context sensitive quick-info which would deliver more comprehensive explanations about the behavior of individual components or search tool alternatives. Moreover the users asked for a textual explanation to come up as soon as they touched one of the input ports of a component to indicate which other component could be connected to this port.

- **Context sensitive behavior**

The select box in which all components available in the system are shown is very complex. Thus during the workshop the users had problems to find the appropriate components.

Collaborative Aspects

During the workshop some users started to discuss how to carry out the tailoring tasks collaboratively. Circumstances requiring searching typically involve time pressure and do not allow time for tailoring for most users. The user providing local support was very enthusiastic about the tailoring environment. During the workshop the users already discussed how the tailoring work could be divided among them. Pointing to her colleague who provides local support, the administrative clerk suggested the following

division of labor: „*The alternatives are good for us. The assembly-mode is for you.*“ Assuming such a division of labor, the colleague providing system support required an additional tool to distribute newly assembled search tools among the users. He argued that his job would become much easier with such a tool.

Exploration

Another issue, which arose during the discussion, was the necessity to explore newly assembled search tools. The user acting as local system support asked to be able to test newly created search tools: „*I need to know whether these things do what they are supposed to do.*“ Nevertheless, the exploration of a tool which involves searching outside the own desk can cause disturbances of the other users. A statement of the secretary made clear that she would carefully select those users who would accept such a disturbance.

Even tolerant colleagues will probably not accept permanent interruptions due to other people's testing. Therefore, a test environment to allow users to explore their newly assembled or selected search tools without affecting other users would probably encourage tailoring. Such a test environment should also cope with the users' worries to break existing artifacts. In that sense, the user providing local support asked for an „undo“-command to be able to recover the old search tool if he should have made a mistake in modifying it.

Summary of Evaluation

The main results can be summarized as follows: The users understand the basic ideas of component-based architectures but at the moment they were not able to tailor their tools and they were even less capable of assembling new tools. This is caused by the complex user interface. Workshop 2 of Search Tool 2 was the source of many new ideas that can help developing a new more easy-to-handle user interface. The other question we pointed out was to explore if the components, we have implemented so far, can be used for designing all the required search tools. The result of the workshop was that we need some additional components but in principle we have implemented the right components.

Search Tool 3: Field Test

Based on the experiences with Search Tool 2 we built Search Tool 3 which not only contained enhanced functionality but was used within the SR for several weeks (cf. Engelskirchen 2000 for a detailed description of all aspects of Search Tool 3).

Setting

For Search Tool 3 the component language and the tailoring environment were extended. In the following we carried out a field study with three users in the SR. Other users of the SR were asked to provide search permission on their documents eventually needed by the participants of the field test.

We presented the new search tool environment in a workshop in which the three users, one user advocate and three designers participated. After this, it was introduced for the field test. In the following two weeks, the users were supported continuously by a user advocate. Also, project members visited each user at least twice for a 60 - 120 minutes time span. During these prearranged visits project members encouraged tailoring activities related to the users' search tasks. The tailoring process and the emerging problems were observed, written notes were taken during the observation and transcribed directly after the visit.

At the end of the observation period, a last extension of the search tool was introduced that included a simulated search in the groupware environment. A few days after installing this new version, we carried out semi-structured interviews with the users. The interviews covered the following issues related to the tailoring environment: patterns of collaborative tailoring, usage of textual documentation (manuals, help functions, annotations), occasions and means to experiment with applications, and further design requirements. The interviews lasted about 60 minutes and were carried out at the users' workplaces. Written notes were taken during the interviews, a transcription was carried out immediately afterwards. Shortly after the interviews we copied all the tailored artifacts for analysis.

Implementation

In the following all extensions of Search Tool 3 are described (cf. Wulf 1999).

- **Some new components**

We have implemented some new components, e.g. more expressive display components and a counter component which is capable of counting the found objects without displaying them. Regarding the privacy discussion this component allows for finding an object without seeing it. Thus one can search for an object only to get to know about its existence.

- **More expressive descriptions of the components**

As described above the users had problems to select the basic components appropriately. These basic components were labeled by rather design-oriented names. Therefore, the users found it difficult to select the appropriate components from a linear list in which the components are itemized. We tackled the problem in three different ways. First, we tried to find more meaningful names for the individual components in collaboration with the users. Second, we added icons to the presentation of the components in the list. These icons resembled the visual presentation of the components at the interface. Third, we classified the components into four different types and used this classification scheme as an additional hierarchy in the toolbox menu. Moreover, we implemented a context sensitive select box which offers only those components which could be connected to the active port.

- **Hypertext-based Help**

We developed a hypertext-based help menu for the search tool window and the toolbox window. The help texts of the tool-box covered all elementary components by a brief explanation of up to six sentences depending on their complexity. Screen shots were added where necessary.

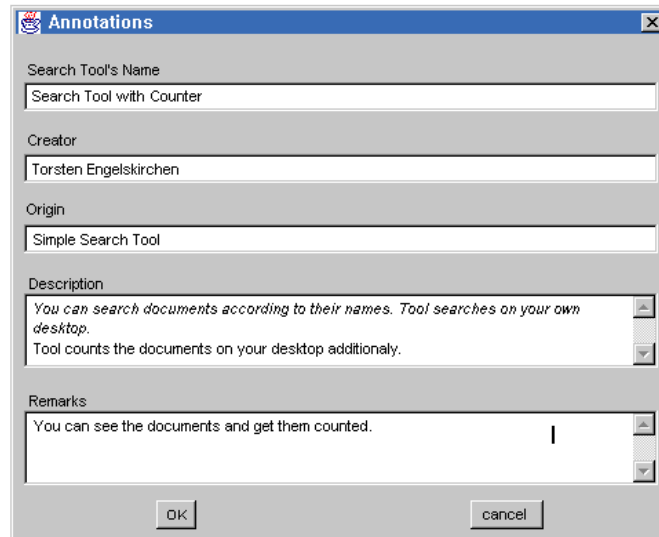
- **Search Token**

For privacy reasons we decided to allow searching only on everyone's own desk unless a person put a special search token in a folder which then allowed others to search this folder. This visible token meets users requirements to be very aware of the fact that someone else can search in one of their folders.

- **Ability to share tailored artifacts**

Search Tool 3 allows users to share tailored artifacts by saving them in a shared workspace. There, they can be deleted, renamed or copied. It is also possible to annotate these artifacts (see below).

- **Annotations: The facility to describe tailored artifacts**



The image shows a Windows-style dialog box titled "Annotations". It has a standard title bar with a close button. The dialog contains several text input fields and two buttons at the bottom. The fields are labeled "Search Tool's Name", "Creator", "Origin", "Description", and "Remarks". The "Search Tool's Name" field contains the text "Search Tool with Counter". The "Creator" field contains "Torsten Engelskirchen". The "Origin" field contains "Simple Search Tool". The "Description" field contains two lines of text: "You can search documents according to their names. Tool searches on your own desktop." and "Tool counts the documents on your desktop additionally.". The "Remarks" field contains the text "You can see the documents and get them counted.". At the bottom of the dialog are two buttons: "OK" and "cancel".

Figure 7: Annotation window

The initial workshop and the field test showed that users are hardly able to deduce the meaning of all components just from their names and the way they are classified. Hence, we generated possibilities to describe the functionality textually. Features which allow to describe components and tailored artifacts have to take the different actors into account who produce this documentation (cf. figure 7).

- **An exploration mode**

For users trying to find out how an unknown search tool works is difficult in a real-life setting when search permissions have to be explicitly granted. Users who try out a new search tool, which unexpectedly does not find any documents on other users' desks, have difficulties to judge whether this outcome is due to a wrong understanding of the tool or missing search-permissions on others' desks. Therefore, we decided to extend the search tool environment by an exploration mode. For privacy reasons the final version of the exploration

mode included the possibility to explore a simulated data space that did not belong to a person but contained artificial data generated particularly to support people in exploring the search tool's functionality.

Figure 8 shows a screen shot of the exploration mode. The window on the top right allows populating the simulated desktops with experimental data. The other two windows (search tool window at the bottom and tool box window on the top left) show the tailoring environment in exploration mode. The windows look exactly like the original ones with the exception of the background color. The search tool presented in the search tool window operates on the experimental data visible in the top right window.

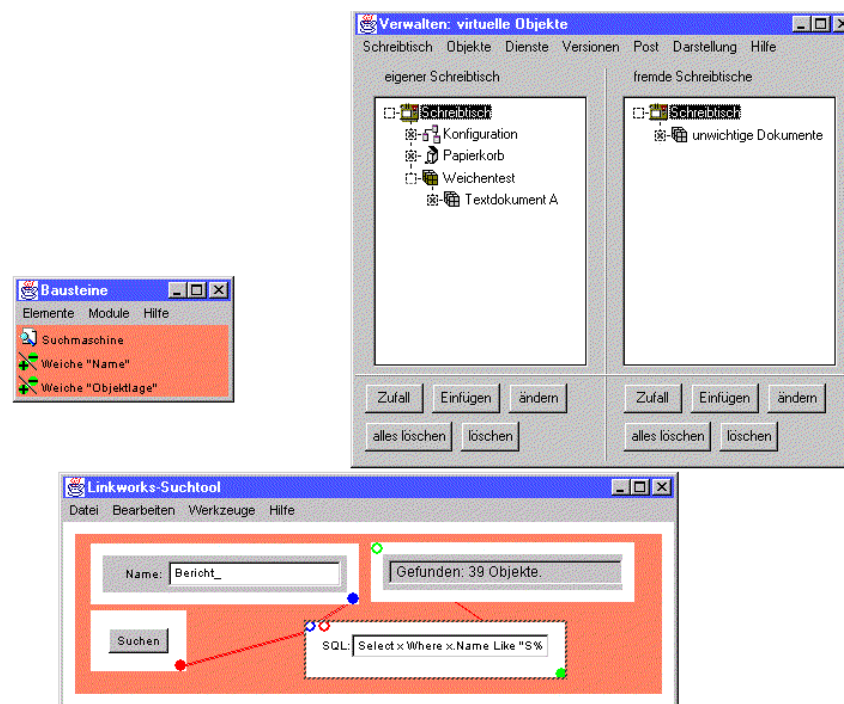


Figure 8: The exploration mode

Evaluation

Structuring Components and Tailored Artifacts

Search Tool 3 included some new components with more expressive descriptions. During the field test we found that these features improved the

ability of the users to select elementary components. Still, when the components were invisible during the search tool's usage or their functionality was rather complex, it turned out to be difficult to communicate their meaning by a name or an icon (i.e.: it was difficult to find appropriate names and icons for switches). Besides, the components' classification scheme which we used in order to establish an additional hierarchy level in the menu was not understood by the all users. So they suggested abandoning the additional level in the hierarchy of the menu and applying it just as a means to structure the linear list. Given a list of all in all 17 elementary components this was a viable solution. Nevertheless, if a component based tailoring language consists of considerably more elementary components, the former approach needs to be pursued, and that may lead to the mentioned problems. A practical approach to solve this problem would be a tailorable menu structure. Yet, if each user could modify the structure individually, this may lead to problems in collaboration.

The naming of tailored artifacts became a problem in the field test, as well. For instance, the clerk from the public relations department used her own convention to name search tools she had modified. This convention was not well understood by the other users. To encourage collaborative use of tailored artifacts common naming conventions are important. The classification of tailored artifacts may lead to further problems. Right now there are just two linear lists for the compound components and the search tool alternatives. These lists are in alphabetical order according to their names. Nevertheless, with an increasing number of these artifacts individually or collectively tailorable classification schemes seem to be indispensable.

Shared Tailored Artifacts and the Division of Labor

During workshop 2 of Search Tool 2 it had become clear that the users liked the idea of being able to share tailored artifacts if there was a local expert responsible for these activities. During the field test the implemented sharing mode was therefore welcomed by the users. The two none-expert users appreciated to be provided with high quality tailored artifacts. The local expert, however, stated in his final interview that he felt a bit uneasy if any tailored artifacts would become publicly available, and thus, other users

could see when and what he tailored. He asked for private stores where he could keep his experimental artifacts.

Annotations and Help

The field test showed that users are hardly able to deduce the meaning of all components just from their names and the way they are classified. Hence, we generated possibilities to describe the functionality textually. Features which allow to describe components and tailored artifacts, have to take the different actors into account who produce this documentation.

As programmers created the elementary components and the tailoring environment, we developed a hypertext-based help menu for the search tool window and the toolbox window. The help texts of the tool box covered all the elementary components by a brief explanation of up to six sentences depending on their complexity. Screen shots were added where necessary.

During the field study it turned out that the local expert was the only one who used the help menu at least sporadically. All users indicated difficulties in finding the access point to activate the help-texts and the location of the desired explanation in the hypertext presentation.

Other than with the predefined elementary components, users generated compound components and full search tools themselves. Thus, the description of these artifacts has to be carried out by them. As the textual documentation of design rationales imposes extra burden and is therefore often omitted (cf. Grudin 1996), we tried to provide as much technical support as possible. We have implemented an annotation window, which consists of five different text fields: „name”, „creator”, „origin”, „description”, and „remarks” (cf. figure 7). The „name” field is automatically marked whenever a tailored artifact is created. In the „creator” field the user who builds a tailored artifact can put in his name. In the „origin” field, a reference is generated automatically in case a tailored artifact has been created by modifying an existing one. In the „description” field the creator should clarify the function of the component. In case a compound component is derived from an existing one the original description is copied automatically and put in *Italics*. The „remark” field can contain further comments. Contrary to the help texts, the annotations were accessible directly from the display of the respective tailored artifact.

In the field test annotations were used more frequently than the help-texts. This result is probably caused by an easier access mode and the richer information structure. The users liked the information structure of the annotation window. The documentation of the creator's name was important to them for four reasons. First, knowing about the creator's typical search tasks helps them to understand the functioning of the tailored artifact. Second, the creator's name is an important information for judging the quality of a tailored artifact. Third, it allows contacting the creator for further information. Fourth, the documentation of his name gives the creator a chance to let the organization know about his efforts. The users found the „origin” field helpful as it recorded parts of the tailoring history, and thus, eased understanding of the functioning of the artifact. The „description” and „remark” fields were perceived being essential to increase understanding and were almost always filled in during the field test. Nevertheless, the way they were filled was often regarded problematic. Especially the usage of abbreviations and uncompleted sentences caused considerable problems. Thus, here again user groups carrying out collaborative tailoring activities need to develop appropriate conventions.

Exploration

Discussing the concept of exploration during the interviews, the user providing local support found the exploration environment useful to test whether a search tool really finds what it was supposed to find. The other clerk was quite reserved towards this concept because to her it seemed too complex to handle. The efforts to create experimental data and to handle the different roles appropriately seemed too high for her compared to the benefits: checking whether a given search tool is doing what it is supposed to do.

While observing users' tailoring habits we found many occasions when building and experimenting interleaves. For instance, users modified a given tool to better understand its functioning and that of some of its parts, or they carried out minor modifications in an existing tool and experimented with that. Therefore, we believe that building tailored artifacts and experimenting with them should be both supported in the exploration mode.

We extended the exploration mode in a way that it became possible to experiment not only with the tailored artifacts but also with the tailoring functions. To build an explorable tailoring environment, we applied the

concepts experimental data, neutral mode (action is not really performed, only explained) and freezing points (defined status of a file that you can return to after experimenting). Whenever a user decides to switch to the exploration mode of the tailoring environment, a new window pops up. It has a specific color of the frame and contains those windows of the tailoring environment which were active before starting the tailoring mode (search tool and tool box). These windows behave regularly except for those functions, which allow to modify the state of the original search tool environment (e.g. store search tool, rename search tool). These functions are put into neutral mode. So the state transition following their execution is not carried out but described textually. The exploration mode comes up with a copy of the search tool which was active before. Playing the role of experimental data for the tailoring environment, this tool can be modified by means of the tailoring functions. If users decide to leave the exploration mode, they are asked whether they want to store or abandon the outcome of their explorative activities. In any case, the users return to the tailoring environment containing the search tool, which was active before they started the exploration mode.

Discussion & Conclusions

In this paper we described the process of developing three consecutive prototypes of a search tool to be used for search in a shared workspace. The process included user participation of different kinds at various steps of the development process. We learned much about the searching for files in groups, the chances and limits of making such a search tool tailorable for individual and group needs and the chances and limits of component architectures to do so. Several insights are gained from the process as a whole. We can now provide some answers to our research questions *How can we design for searching in groupware?* and *How well are components suited for runtime tailorability?*

Designing for searching in groupware encompasses several dimensions. In the *process* of design the combination of having users participate and developing several versions of the search tool evolutionarily proved to be effective: Each workshop, interview and evaluation provided us with insights that could not have been won by top-down one-step design. Since we not only wanted to support particular organizations but also wanted to show the general feasibility of our approach the process was rather costly

using much time and personnel. However, since we provided evidence that our approach was effective subsequent endeavors to construct tailorable software with limited functionality can also be efficient. Nevertheless they still have to bear in mind that providing a flexible organization with adequate software support is always a laborious and ongoing effort.

Search Tool 3 as the final *product* of the design process included much functionality, all of which was implemented due to requirements from users or improvements derived from literature survey. The resulting complexity led to a differentiation in using the functions:

- few of the functions are used frequently by many users, e.g. looking for or trying out new search tools from the search tool pool;
- some are used sometimes by many users, e.g. exploration of an unknown search tool in case a task or a group changes or the written explanation of the search tool is not sufficient;
- some others again are only used by few people, e.g. the possibility to create new search tools from scratch or perform severe modifications on an existing search tool.

This is encouraging in two ways. Firstly, the fact that the functions were used at all and that there were differentiated patterns of usage proved that there really was a need for a tailorable search tool. Secondly, and more important, it shows that the functionality we added resulted in more than just the sum of the parts: the different options and levels of using, annotating, exploring, modifying and constructing search tools led to a variety of options for users to tailor. Thus, they could move up to a plateau of the „tailorability mountain” (MacLean et al. 1990) that suits their abilities and needs with the option to engage in more tailoring activity by talking to a local expert. This was also supported by our layered architecture which allowed for the assembly of search tools on different levels of granularity and in fact supported the evolution of a local expert.

There are two ways in which the fact is taken into account that the search tool was meant to work in a collaborative setting to find files in a shared workspace. On one hand the participating users were the *objects* of the search activities of others. Therefore there was a request for mechanisms to protect privacy. To meet this request we implemented an option to find objects without being able to display them, a search token to put into a

folder to allow this folder to be searched, and the possibility to explore a simulated data space rather than the real shared workspace to learn how a search tool works. On the other hand the users were *subjects* of the search. Not only did they require efficient mechanisms for their individual search but there was also the need to provide mechanisms for collaboration regarding search activities. Therefore, our search tool environment provided mechanisms to share, distribute and annotate search tools for others to use or modify, thus supporting weak forms of collaboration. The workshops show that stronger forms of collaboration where users meet and discuss the tailoring of a search tool are generally supported by the possibility to tailor and particularly by the runtime environment, where ideas that have been discussed can easily be turned into a tailored search tool and be tested. We assume that such a tool for easy tailoring may serve as a medium that encourages groups to discuss group standards that then can be shared. The systematization of customizations resulting from a collaborative tailoring process would then contribute to common norms and conventions needed for collaborative work. Again, one must bear in mind that this eventually rewarding activity of collaborative tailoring requires willingness and patience.

Sharing tailored search tools seems most helpful for small work groups with a rather similar work context. Being able to explore functionality in groupware context with real or simulated data on the other hand becomes more important the more users are involved: The complex interdependencies of different users and their privacy and other settings require more than a textual description of functionality. However, this rather complex form of supporting users' understanding must always be accompanied by simpler forms like tutorials, context-sensitive help texts and a colleague to ask.

Our specific research focus was on how well components are suited for runtime tailorability. For quite a while it has been demanded to provide tailorability not only on a surface level but to embed it deeper into systems. Using a component architecture to realize runtime tailorability proved to be a good choice. The process of developing a search tool showed that using a component architecture suited for runtime tailorability is a feasible way to meet multiple requirements for such a search tool. The relative simplicity of the task consisting of input, searching, and output made it possible to identify an adequate level of decomposition and the relevant components. For more complex tasks the identification of a good decomposition will be

much more difficult but, as we believe, also possible provided that there is some guidance for users and some time for them to have experts and solutions evolve. The possibility to represent the connections visually provided for a significant reduction of complexity. Considering the difficulties that some users had in assembling a search tool in our workshops and during the field test we doubt that other forms of modularization (e. g. blocks of program code) would have been manageable to all but very few skilled users. Being able to modify components and their wiring during runtime was vital for the acceptance of the search tool construction set. If there had not been such a tight integration between using and tailoring the search tool, the threshold to create, modify or test a search tool would have been critically high.

There is also evidence that such a component-based approach can be enhanced and possibly be used for the design of a variety of systems. The strict separation between search tool specific components and the design of the runtime environment makes it comparatively easy to enhance the prototypes by further components. Therefore we think that the design and runtime environment can easily be reused: it is technically possible to create a multitude of different tailorable tools and applications. However, considering the simple structure of a search tool it remains an open question where the creation of different kinds of complex tailorable software with these forms of components is no longer reasonable.

The process described above and the lessons learned encourage us to continue research that combines theoretical and empirical evidence to come up with suggestions that can make a difference to software designers, introducers, and users.

References

- Barreau, Deborah; Nardi, Bonnie A. (1995): *Finding and Reminding: File organization from the Desktop*. In: SIGCHI Bulletin July 1995, Vol. 27 (3). pp. 39-43.
- Bentley, Richard; Dourish, Paul (1995): *Medium vs. Mechanism: Supporting Collaboration Through Customization*. In: Proceedings of ECSCW '95. Kluwer. pp. 133-148.

- Carter, Kathleen; Henderson, Austin (1990): *Tailoring Culture*. In: Reports on Computer Science and Mathematics no. 107, Åbo Akademi University 1990. Proceedings of 13th Information Systems Research Seminar in Scandinavia (IRIS). pp. 103-116.
- Engelskirchen, Torsten (2000): *Explorationsunterstützung in Groupware am Beispiel eines anpaßbaren Suchwerkzeugs (English: Supporting Exploration in Groupware: the Example of a Tailorable Search Tool)*. Department of Computer Science III, University of Bonn. Diploma Thesis.
- Henderson, Austin; Kyng, Morten (1991): *There's No Place Like Home: Continuing Design in Use*. In Greenbaum, Joan; Kyng, Morten: *Design at Work - Cooperative Design of Computer Systems*. Hillsdale, NJ, Lawrence Erlbaum Associates. pp. 219-240.
- JavaSoft (1997): *JAVABEANS 1.0 API Specification*. Mountain View, California, SUN Microsystems.
- JCSCW (2000): *JCSCW - Computer Supported Cooperative Work (CSCW): The Journal of Collaborative Computing*. Vol. 9 (1). Special Issue on Tailorable Systems and Cooperative Work.
- Kahler, Helge (1996): *Developing Groupware with Evolution and Participation - A Case Study*. In: Proceedings of PDC '96. Computer Professionals for Social Responsibility. pp. 173-182.
- Klößner, Konrad; Mambrey, Peter; Sohlenkamp, Markus; Prinz, Wolfgang; Fuchs, Ludwin; Kolvenbach, Sabine; Pankoke-Babatz, Uta; Syri, Anja (1995): *POLITeam: Bridging the Gap between Bonn and Berlin for and with the Users*. In: Proceedings of ECSCW '95. pp. 17-31.
- Mackay, Wendy E. (1990): *Patterns of Sharing Customizable Software*. In: Proceedings of CSCW '90. pp. 209-221.
- MacLean, Allan; Carter, Kathleen; Lövstrand, Lennart; Moran, Thomas (1990): *User-Tailorable Systems: Pressing the Issues with Buttons*. In: Proceedings of CHI 90. pp. 175-182.
- Malone, Thomas W. (1983): *How do People organize their Desks?* In: ACM Transactions on Office Information Systems, Vol. 1 (1). pp. 99-112.
- Mambrey, Peter; Mark, Gloria; Pankoke-Babatz, Uta (1996): *Integrating user advocacy into participatory design: The designers' perspective*.

- In: Proceedings of PDC '96. Computer Professionals for Social Responsibility. pp. 251-259.
- Mørch, Anders I. (1997): *Method and Tools for Tailoring of Object-oriented Applications: An Evolving Artifacts Approach*. University of Oslo, Department of Informatics. Dr. Scient. Thesis Research Report 241.
- Nardi, Bonnie M. (1993): *A Small Matter of Programming*. Cambridge, Massachusetts, MIT Press.
- Prinz, Wolfgang; Kolvenbach, Sabine (1996): *Support for workflows in a ministerial environment*. In: Proceedings of CSCW '96. ACM Press. pp. 199-208.
- Rao, Ramana; Card, Stuart; Johnson, Walter; Klotz, Leigh; Trigg, Randy (1994): *Protofoil: Storing and Finding the Information Worker's Paper Documents in an Electronic File Cabinet*. In: Proceedings of CHI '94. ACM Press. pp. 180-185.
- Stiemerling, Oliver (1997): *Supporting Tailorability in Groupware through Component Architectures*. In: Proceedings of ECSCW '97 Workshop on Object Oriented Groupware Platforms. pp. 53-57.
- Stiemerling, Oliver; Cremers, Armin B. (1998): *Tailorable Component Architectures for CSCW-Systems*. In: Proceedings of 6th Euromicro Workshop on Parallel and Distributed Programming. IEEE Press. pp. 302-308.
- Stiemerling, Oliver; Kahler, Helge; Wulf, Volker (1997): *How to make software softer - designing tailorable applications*. In: Proceedings of DIS '97. pp. 365-376.
- Suchman, Lucy; Wynn, Eleanor (1984): *Procedures and Problems in the Office*. In: Office Technology and People, Vol. 2. pp. 133-154.
- Szyperski, Clemens; Pfister, Cuno (1996): *Component-Oriented Programming: WCOP '96 Workshop Report*. In: Proceedings of 10th European Conference on Object-Oriented Programming ECOOP '96. pp. 127-130.
- Trigg, Randall; Bødker, Susanne (1994): *From Implementation to Design: Tailoring and the Emergence of Systematization in CSCW*. In: Proceedings of CSCW '94. pp. 45-54.

- Trigg, Randall H. (1992): *Participatory Design meets the MOP. Accountability in the design of tailorable computer systems*. In: Proceedings of 15th Information Systems Research Seminar in Scandinavia (IRIS). pp. 643-656.
- Won, Markus (1998): *Komponentenbasierte Anpaßbarkeit von Groupware (English: Component-Based Tailorability of Groupware)*. Department of Computer Science III, University of Bonn. Diploma Thesis.
- Wulf, Volker (1997): *Storing and Retrieving Documents in a Shared Workspace: Experiences from the Political Administration*. In: Proceedings of Human Computer Interaction: INTERACT 97. Chapman & Hall. pp. 469-476.
- Wulf, Volker (1999): *“Let’s see your Search-Tool!” - Collaborative use of Tailored Artifacts in Groupware*. In: Proceedings of Group 97. pp. 50-59.

Seventh Paper

Collaborative Tailoring – Eight Suggestions

Introduction

In the past fifteen years the computer has found its way into many homes and all offices. Other than in the early days of electronic computers most of the software used is no longer programmed by those who use it. Moreover, in the past five years computers have been increasingly connected to allow for data exchange, communication and collaboration over a network. Thus, the task to support users in what they are doing has become increasingly complex. One of the ways to deal with this complexity is to continue design in use.

This may include efforts to bring designers and programmers of computer software to people's work places or homes to understand their requirements, then go back to their sites and change the software adequately. Additionally, for smaller changes it is obvious to consider tailorability, i. e. the possibility to modify the functionality of technology while the technology is in use in the field. However, this makes it necessary that the software can be adequately changed by users. To ensure this, not only must the software be changeable at all, but also those changeable parts should cover the relevant aspects of the users' work task including the way they work together.

For users whose work task includes collaboration with others the possibility to tailor a software to their personal or group need includes the hope for increased efficiency and improved collaboration. Eight suggestions of how collaborative tailoring can be supported are presented in the following. They are the result of literature study and own work on two cases of collaborative tailoring.

In the next section, related work on tailoring and collaborative tailoring is presented. After this, the two cases are briefly introduced. Then, the eight suggestions for collaborative tailoring are proposed.

Related Work

In this section the main relevant literature for approaching collaborative aspects of tailoring is presented. A large body of work has been devoted to theoretical and empirical work on how tailoring and particularly collaborative tailoring of software could be supported technically, why it should be supported and how software has been tailored by users. Some authors also described systems they implemented and that were sometimes evaluated. The presented literature is restricted to contributions explicitly discussing tailorability as a major issue.

Tailoring Software

Modern software usually comes with a cornucopia of features, that permits many forms of file creation or modification. However, this abundance very often makes it difficult for users to find out how to perform a particular task with the software or how to do so efficiently. *Tailoring* is the technical and human art of modifying the functionality of technology while the technology is in use in the field. Consequently, the feature of a software to be tailored is called *tailorability*. It is widely agreed that tailorability is one of the major future challenges in the design of interactive systems (Bentley & Dourish 1995, JCSCW Vol. 9 (1), Special Issue on Tailorable Systems and Cooperative Work).

Reasons for Tailoring

The main and overall reason why software should be tailorable and needs to be tailored is the *complexity* of the setting where it is used and of the task it is used for. Trigg (1992) provides three main reasons why systems should be tailorable:

- The *diversity* along several dimensions like persons, tasks or objects of tailoring must be taken into account when selling a generic software that is supposed to fit different settings. A lawyer in Saudi Arabia probably uses a word processor differently than a researcher in Europe. This even

holds true for custom-made software: people may work in the same office with different tasks or even only different usage patterns of mouse and keyboard. And also one single person may want to perform a single task differently at different times or related to a different context.

- The *dynamism* (called *fluidity* by Trigg) of individual and organizational work corresponding to the changing nature of work over time requires software to also change over time. The structures of work organization and collaboration may vary considerably in relatively short time periods.
- The *uncertainty and ambiguity* about the exact work practices and procedures even in the perception of the workers makes it necessary to leave room for alternative ways of performing tasks. Trigg (1992) reports a case where a person in an interview about her work practice told stories that revealed uncertainties of her own view of her work situation, e.g. how reasonable it was to archive certain documents.

Grudin (1991) hints at the fact that software is almost never programmed to be used by one person at one time but by many users at many times who are often not personally known to the programmers or who perform a task unknown to the programmers.

Haaks (1992) distinguishes different dimensions of tailoring including initiator and actor, object, aim, time, and scope of validity. The *initiator and actor* can be the system or a user. The *object* of tailoring depends on the taken perspective and can range from setting default values via limiting the available functionality to ease the learning of the system to modification and enhancement of functionality. The *time* when a system is tailored can also differ. Tailoring can take place before the first use of the system, between phases of use and during use. Haaks concedes that only tailoring a system before the first use is never enough since it does not take into account the dynamism of use. He describes the *scope of validity* as depending on the concerned group of persons (a single user, a group of users or all users), the time span (a session or a phase of usage) and the affected aspects of the software.

Definitions

There are different terms, concepts and taxonomies connected with the idea of users modifying software during use, among them *individualization*, *personalization*, *adaptation*, *customization*, *end-user modification*, and

tailoring. More than 20 years ago the EMACS editor provided mechanisms for extension by the user while it was running (Stallman 1981, p. 149):

“Many minor extensions can be done without any programming. these are called customizations, and are very useful by themselves.”

Some years later a more thorough distinction is provided by Trigg et al. (1987, p. 723) who described Xerox PARC’s information structuring system NoteCards:

- *“a system is flexible if it provides generic objects and behaviors that can be interpreted and used differently by different users for different tasks*
- *a system is parameterized if it offers a range of alternative behaviors a user can choose among*
- *a system is integratable if it can be interfaced to and integrated with other facilities within its environment as well as connected to remote facilities*
- *a system is tailorable if it allows users to change the system itself, say, by building accelerators, specializing behavior, or adding functionality”*

The authors stress the importance of tailorability since it is the one of these features that allows users to change the system behavior in ways unanticipated by the system’s designers.

Mørch (1995a) defines *tailoring* as the adaption of generic software applications to the specific work routines of a user organization. Based on a literature survey he identifies three levels of tailoring:

- *Customization* means selecting among a set of pre-defined configuration options by direct interaction or setting parameters;
- *Integration* can be *hard integration* where a component is attached physically to the application or *soft integration* where a component is integrated by means of a macro, script, or agent;
- *Extension* means adding new code to the application.

Similarly, Henderson & Kyng (1991) distinguish the three levels choosing between alternative anticipated behaviors, constructing new behaviors from existing pieces, and altering the artifact. They consider these to be activities

that the tailors must do that relate to the above mentioned four properties of systems by Trigg et al. (1987) and just take different perspectives on the same issue.

Henderson (1997, p. 1) defines

“Tailoring is the technical and human art of modifying the functionality of technology while the technology is in use in the field.”

This definition is adequately broad and includes both the relevant technical and socio-organizational aspects. It also stresses the importance of really using a technology at some place and in particular work settings in order to be able to know how it should be tailored there. Therefore, this definition is also used for the report in hand.

Tailoring, Using, and Developing

Tailoring software can be distinguished from use and development although it bears similarities with both. On one hand it is a way to continue design in use to account for unanticipated needs, on the other hand it extends use by providing means to make it effective and efficient. Henderson & Kyng (1991) argue with the relative stability of an application in claiming that people tailor when they change stable aspects of an artifact. However, they also admit that the distinction may be difficult: Changing the font of a document can be considered to be use or tailoring. They also introduce the notions of *subject matter* vs. *tool* of work and claim that changing the subject matter is use while changing the tool is tailoring. Again, the distinction is not always clear, since one person's subject matter is another person's tool: For a person using an application programmed in JAVA, this application is a tool, whereas for its programmer it is the subject matter, and the JAVA compiler is the tool (and for the compiler builders it is the subject matter). Finally, if the effect of a modification is immediate only the action can be considered to be use.

Collaborative Aspects of Tailoring

Since more and more work with a computer is done in groups where people work on similar or the same tasks and files tailoring very often has collaborative aspects to it.

In the literature on tailorability collaborative aspects have so far not played a major role but have been mentioned at several occasions. Already Stallman (1981) reports that users not only think of small changes and try them, but also give them to other users. Mackay (1990) researched how people actively shared their tailoring files with each other. The study was conducted at two research sites. At the first site 18 people using Information Lens (Malone et al. 1988) to tailor the management of their emails were observed over a period of three or more months. This included several interviews per participant and the collection of automatically gathered data. Mackay reports that several people shared Information Lens rules (i.e. text files containing information about the filtering of mails) including two manager-secretary teams who used the rules to support a standard form of communication. At the second site a group of 51 users on a common project sharing Unix tailoring files were observed over a period of four months. The data gathered stem from one or more interviews per user and also included copies of their tailoring files. More than three-quarters of the participants received tailoring files from others since they had joined the project. The following methods to obtain or give tailoring files or ideas were identified (p. 213):

- *“Someone helps you to get set up.*
- *You ask someone to help you get set up.*
- *You get the standard system file and use it.*
- *You have a problem and ask someone for help.*
- *New ideas are posted electronically in a common area and you look.*
- *Someone has a new idea and tells you about it.*
- *Someone tells you to look in the common area.*
- *You have a symbolic link to someone else’s file which is automatically updated.*
- *You walk by and see someone else’s screen and ask how something was done.*
- *You watch someone performing some task, notice a useful technique, and ask, how it’s done.*
- *You help a newcomer get setup with a version of your files.*
- *You post an idea in the common area.*

- *You tell your friends about a new idea.”*

Depending on the job category (e.g. Manager, Secretary or Application Programmer) the different groups borrow and lend files with different intensity and have a different percentage (0% to 38%) of *translators*. To Mackay these are persons who actively share their files and talk to the recipients of the files. She concludes both cases by criticizing that staff members are often not rewarded for sharing tailoring files and requests that tailorable software should provide the ability to browse through others' useful ideas and that it should include better mechanisms for sharing customizations which then may serve to establish technical or procedural standard patterns.

The role of a local expert was also highlighted by Gantt & Nardi (1992) who describe what they call *patterns of cooperation among CAD users*. They studied the use of a Computer Aided Design (CAD) system by conducting in-depth interviews with 24 informants and collecting and analyzing the informants' CAD artifacts. They distinguish between *local developers* who write macros, programs and scripts and help end users in tailoring on one hand and on the other hand *gardeners* as a sub-group of local developers. With gardeners the informal position of a local developer has evolved into a formal or semi-formal position. They are responsible for writing and disseminating standard macros and programs at the corporate and department level. Usually, a gardener has both domain and computer knowledge and often starts from the domain side and then acquires the necessary computer expertise. Gantt & Nardi support the contention that the activities of local experts should be recognized and promoted since a local expert and particularly a gardener can save the organization's time and money by offering valuable resources like macros and programs to the entire group. They admit, however, that it may be difficult to find a person with the right combination of technical and social skills.

Nardi & Miller (1991) report that spreadsheets offer strong support for cooperative development of applications. They present results from an in-depth-study based on data of 350 pages of transcriptions from interviews with 11 users of several spreadsheet products. Nardi & Miller conclude that spreadsheet co-development is the rule rather than the exception and that spreadsheets support the sharing of both programming and domain expertise. In their study they describe, how spreadsheet users (p. 163)

- *“share programming expertise through exchange of code;*
- *transfer domain knowledge via spreadsheet templates and the direct editing of spreadsheets;*
- *debug spreadsheets cooperatively;*
- *use spreadsheets for cooperative work in meetings and other group settings; and*
- *train each other in new spreadsheet techniques.”*

They identify the three kinds of users *non-programmers*, *local developers*, and *programmers* where the first group is responsible for most of the development of a spreadsheet, and the second and third group contribute code to the spreadsheets of less experienced users. Local developers are technically less experienced than programmers but serve as consultants for non-programmers in their work environment. For spreadsheet applications it can be argued that using and tailoring them are closer together than for many other applications, since their usage in the sense that you just put in numbers and calculate something implies the prior work of defining code behind the spreadsheet's cells which is responsible for the calculation. This is usually done by persons who have domain knowledge and, as Nardi & Miller note, usually done cooperatively. Considering the fact that more and more off-the-shelf software needs tailoring and offers mechanisms for it the presented results encourage the tighter integration of using and tailoring.

Trigg & Bødker (1994) found an emerging systematization of collaborative tailoring efforts in a government agency. In their study they were looking at the tailoring of word processors performed by four persons in a Danish administration over a year. After this they conducted hour-long interviews. The four protagonists work on the borders between technology development and everyday work, two of them being officially recognized local developers whose tailoring work is part of their job description. The third one is a system supporter and the fourth person is the least technologically inclined of them. Tailoring at their organization mainly means customizing the word processors button panels, macros, and standard forms. Trigg & Bødker explicitly distinguish tailoring from programming since the latter moves from analysis to design to realization while the former basically consists of trial and error where the starting point often is a personal solution that may become more stable and then used by several people after a constructive process of small improvements. They observed that tailoring

is often a collaborative process where the idea and the basic work is performed by the local developers who then pass on their partial solution to the programmer for improvement. Also, the learning of tailors has distinctly collaborative character where they ask each other and consider themselves to be on a learning staircase trying to move upwards. The sharing of tailoring files had over the time evolved from an opportunistic spreading where someone heard about tailoring done by a colleague and copied their tailoring files to a more systematic activity: ideas are conveyed to the local developers or the programmer who then implement them. The new tailoring files are downloaded when a computer is rebooted (usually each morning). In particular cases the workers are notified about how they are supposed to use the new functionality. They are also free to ignore the tailoring files. While it is often argued that tailoring leads to an unmanageable abundance of individualized solutions, several aspects imply that tailoring in this organization does rather have a standardizing effect. Standards of particular text blocks and of macros and button panels that reflect the work practice can be developed and widely used because the organization explicitly supports individual and collaborative tailoring and the distribution of tailored files.

Wasserschaff & Bentley (1997) describe how they supported collaboration through tailoring by enhancing the BSCW Shared Workspace system, which is an extension to a standard web server providing basic facilities for collaborative work including information sharing, document management, and event logging and notification. They designed multi-user interfaces for the BSCW system which allow users to take a certain view on the data in the shared workspace. These *Tviews* can be added to the shared workspace as objects in their own right so others can take them, use them, and modify them in the same way as documents and folders. However, there are no evaluation data to support the notion of Wasserschaff & Bentley that *Tviews* can bridge the gulf that often separates the tailoring of surface and presentation features from the deeper aspects of system behavior nor data on actual collaborative tailoring performed around *Tviews*.

As can be seen from the aforementioned examples collaborative tailoring does very often not occur among groupware users but also in groups of users using the same software and thus being able to employ the fact that this software is tailorable and that tailoring files may be exchangeable. Particularly the fact that more and more computers are connected to a local or wide area network creates the infrastructure to exchange tailoring files

even of single user applications easily through the network. Therefore, the boundaries between collaborative tailoring of a single user software and a groupware become fuzzy.

Collaboration certainly takes place where the whole group is actively tailoring. A weaker form of collaboration takes place where an individual, often a local expert, tailors for a group. Certainly, tailoring is more efficient in this case than if all individuals would do it on their own. Also, the tailor and the group share the common goal of improving a system to meet their needs. Usually, as has been described in several of the studies mentioned above, there is also a form of interdependence involved. On one hand, the group depends on the tailor to deliver something that meets their needs, and the tailor needs to know the domain and the requirements from the group. On the other hand, since tailoring is often a constructive and cyclic process the tailor usually depends on the groups feedback to improve the tailoring file. The border for individual actors tailoring for themselves and tailoring for the group is often blurry. Even if a person only tailors for her- or himself someone else might notice a difference to her or his own system and ask how the tailoring was done and if they can have the tailoring file. Also, an individual might tailor something for her- or himself and then think that it might be useful for two colleagues and send it to them or even for all colleagues and put it into a shared workspace. This, in turn, may lead to a second person changing the tailoring file and then again sharing it with others which could lead to numerous versions and a discussion about them. Thus, even if originally there may have been no intention to interact or collaborate, the potential for both is there as soon as an individual tailors.

Two Cases

In the course of working on issues of collaborative tailoring two cases that I worked on shed a different light on tailoring and collaborative aspects of tailoring.

In the word processor case (see Kahler 2001) the explicit aim was to support collaborative tailoring of a single user application mainly by the objectification of tailoring files and by providing mechanisms to distribute and share these tailoring files. To do this an empirical field-study on the collaborative tailoring habits of users of a particular word processor was carried out. Based on these and literature research an extension to this word

processor was developed which provided a public and a private repository for adaptations as well as a mailing function for users to exchange tailoring files. Some notification and annotation mechanisms were also provided. Results of two forms of evaluation indicate that users of different levels of qualification are able to handle the tool and consider it a relevant alternative to existing mailing mechanisms.

In the groupware search tool case (see Kahler et al. 2001) a strong user involvement over time led from the development of an improved untailorable search tool to a tailorable search tool. The design of this groupware search tool was based on the assumption that a useful generic search tool must be highly tailorable. We achieved tailorability by applying an innovative software architecture which allowed to assemble components during runtime. In order to understand how people search in shared workspaces and to support the design we employed interviews and workshops with users as well as a field test to understand users' needs. During the design process we developed a series of prototypes which were then evaluated by office workers. Consequently, the process described and the lessons learned extend from searching in files as a case via tailorability of software as an answer to the resulting requirements to component architecture as a way to implement this tailorability.

Concerning collaborative tailoring both cases complemented each other and the literature: The interviews in the word processor case clearly showed that even for single user software like work processors collaborative tailoring does exist, some of which is not even supported by a computer network, e. g. looking how something is tailored and trying to repeat it at home, or passing floppy disks. The implementation showed the general possibility to enhance already existing tailoring possibilities by features for collaboration using the existing local area network. The laboratory evaluation provided evidence that the concept of sharing and sending tailoring files in analogy to other files can be understood and that it may be useful for people's work. In the groupware search tool case the dynamism of a real organization was better captured so that the practical use including organizational structures like the division of labor between the local expert and the other users became clearer. Together with the literature these two cases provide the material for suggestions to support collaborative tailoring,

Suggestions for Supporting Collaborative Tailoring

This section presents suggestions for supporting collaborative tailoring. They are the quintessence of my work on collaborative tailoring in the last years and my contribution to the ongoing discussion. This section brings together an intensive literature study and collection of descriptions of related work, my experience as action researcher particularly with tailoring and collaborative tailoring, and the identification and implementation of identified relevant features in software and their evaluation in laboratory settings and identifies the most relevant means by which collaborative tailoring can be supported. The suggestions for collaborative tailoring aim at different aspects of collaboration, sometimes showing all or some of the background they originally came from (e.g. individual or joint file organization for suggestions 3 and 6, CSCW for suggestion 4, or socio-technical considerations from the information systems field for suggestion 8).

Note that one of the preconditions for collaborative tailoring is that a system can be tailored at all. There is a large body of literature on the question of how tailoring of a single user application can be supported (see e.g. Oppermann 1994). Some of the following suggestions are inspired by this. However, tailorability of the application underlying the collaborative tailoring efforts is assumed. Another source of inspiration is the discussion of computer supported collaborative work (CSCW) in general. The suggestions presented below focus this discussion to the particular topic of tailoring. Undoubtedly, CSCW as a research field will be a constant source for improved and new suggestions for supporting collaborative tailoring. The suggestions may serve for researchers to refine and transfer to other areas, for software developers to have a guideline for implementing necessary functionality and provide adequate software structures, and for practitioners to be able to select and tailor generic software and to provide organizational structures that support collaborative tailoring. The suggestions reflect that most research including my own cases deals with comparatively small groups that tailor collaboratively, usually not more than ten persons. For larger groups more thought must be given to the scalability and particularly the organizational issues and the danger of decreasing individual involvement going along with a larger group size.

The following suggestions concern both technical and socio-organizational aspects of collaborative tailoring. It is important to note, that neither technical nor socio-organizational measures to support collaborative tailoring alone suffice: they must be combined. Often the distinction is fuzzy and a particular suggestion belongs both areas to a certain extent. In the following, first the suggestions that are located more in the technical area, after this, socio-organizational measures are proposed.

Suggestion 1: Provide Objectification

A prerequisite for most forms of collaborative tailoring is that tailoring is made persistent in tailoring files. The tailoring file contains information that allows changing aspects of a software in use. A well known example of a tailoring file is an initialization (.ini) file of a software that may contain information about the appearance of the software at startup time like window position or colors, a user's profile, file extensions connected to the application, and much more.

This *objectification* (Henderson & Kyng 1991, p. 232) allows users or administrators to access, modify or share the tailoring files by the usual means that files are processed. Generally, .ini files can be modified and saved with a text editor. Very often the application that the .ini file belongs to, changes the .ini file when a user changes preferences or options within the application.

Thus, the already existing infrastructure for file processing like the operating system, the network and applications like text editors can be used.

Some operating systems provide a common file where information including tailoring information for many files may be stored (e. g. Microsoft Windows NT's registry). Such a centralized approach has both advantages and disadvantages. The question arises which information should be kept together.

- all tailoring information of only one application can be in one file: this makes the exchange of tailoring information about a single application easy, but makes it difficult to share tailoring information across applications. The latter may be interesting for macros that may insert certain text blocks, or if some personal information changes that should be changed for all applications, e. g. a login name.

- all tailoring information about many files can be in one file: this may make it difficult to extract particular tailoring information about one file to share it with someone else (see suggestion 2: *Allow Sharing of Tailoring Files*). However, it could be easier to tailor a whole group of applications if necessary.

Which solution is appropriate (if there is a choice at all) depends mainly on two things. Firstly, it is important to provide mechanisms to manipulate the tailoring information. If all tailoring information of only one application is in one file the application usually provides mechanisms to modify this information from within the application by changing settings in a preference or options menu or recording macros or the like. If all tailoring information about many files is in one file and there shall be the advantage of changing it for several applications at a time an extra application must be provided to do so. e.g. for Microsoft Windows NT's registry this is not the case: the registry entries are supposed to be modified by single applications which only modify the entries relevant for themselves. It is difficult and dangerous to the operating systems stability to try to change information about several applications at a time e.g. by a registry editor. Secondly, it may be helpful that the way the tailoring objectification is organized reflects the organizational structure: If there is a strong administrator responsible for application updates and keeping certain structures and standards a central file with all tailoring information about many files may be appropriate. If most users administer much of their own workstations and have their own files structure then a decentralized approach for keeping tailoring information may be more appropriate.

Objectification may allow users an easier technical approach to the tailoring files. It also may have the psychological advantage of moving tailoring closer to using. This requires an integration of the options to modify tailoring files into an application in a way that brings modifying the tailoring files close to using the application for a primary work task. For many users tailoring is something they have to do on top of their primary work. But if tailoring does not mean to change some strange entry in a file with a strange format by means of a hex editor but loading a file into an application and then changing it and saving it similar to the work on an "ordinary" file, the psychological barrier to tailor becomes lower. This is certainly the case for templates that can be created and modified exactly like the respective primary work files but have a less ephemeral character. Thus, objectification can help to overcome what Mackay (1991) identified as the

topmost technological barrier for tailoring, namely that the software was too hard to modify (cited by 33% of the users).

Objectification particularly supports those forms of collaborative tailoring where the tailor and the person for whom something is tailored are different. In this case the objectified tailoring activity can be easily exchanged by the usual means for file exchange provided by an operating system and a network.

Note, that objectification is usually but not necessarily the prerequisite for different forms and modes of tailoring:

- a tailoring activity to a word processor that does not allow an objectification of the tailoring activity in a file can be repeated on a second computer on request of an interested colleague;
- a groupware that allows tailoring but not an objectification of the tailoring activity in a file still permits collaborative tailoring e.g. where several persons tailor the groupware with effects for the whole group;
- tailoring activities may also occur or be exchanged by means of a network in other forms than files, e.g. in a tailoring stream sent and received through ports at specified locations and occasions.

In the word processor case tool bars and document templates including macros were the two forms of tailoring activities that were objectified in files. In the groupware search tool case the objectification was more refined. The layered component based approach enabled us to have components of different granularity (atomic, composed, complete search tools) as objectified tailoring files.

Suggestion 2: Allow Sharing of Tailoring Files

Sharing tailoring files or ideas with others is one of the most common and powerful forms of collaborative tailoring. The interviews in the word processor case (see Kahler 2001) showed how important and common sharing of tailoring files is as a form of collaborative tailoring. The interviewees reported various sharing activities ranging from the exchange of a floppy disk with a tailoring file on it to blackboard mechanisms where a tailoring file was posted for everyone to access.

Sharing itself may not be considered to be a collaborative activity. However, sharing a tailoring file is one basis for asynchronous forms of collaboration, e.g. a person modifying someone else's tailoring file for her own use or a number of successive versions of a tailoring file produced and improved by several persons.

Sharing tailoring files is often supported in either or both of two ways: Either a shared workspace is provided to keep tailoring files for a group to retrieve, or mechanisms for sending or receiving tailoring files are provided. Both basic mechanisms rely on the existence on objectified tailoring files as artifacts to share or send. In analogy to the treatment of other files that are worked on collaboratively and depending on the concrete task and organizational setting both forms have advantages. If there are already shared workspaces for other files existing then a particular shared workspace for tailoring files can be very useful, because the group is used to working with shared workspaces and the existing infrastructure for sharing files can be used to share tailoring files. Also, there is less trouble with different versions of a file since all persons have access to the same (and possibly newest) version of a tailoring file. A shared workspace also represents the idea of a group working together better and contains more synergetic potential. A similar form of sharing could be provided by links to an original tailoring file that is stored e.g. in the author's electronic space. If other persons link to this particular tailoring file they all work with the same version at a time. This can be considered a distributed shared workspace. Like in a centralized shared workspace access rights ensure who can maintain and update a tailoring file. For these actions some administrative effort may be required (see suggestion 7: *Make Administration and Coordination Easy*) and awareness about changes (see suggestion 4: *Provide Awareness of Tailoring Activities*) and some annotation about what made the change necessary or what it does (see suggestion 5: *Make Annotations and Automatic Descriptions Possible*) are helpful.

The approach of sending and receiving tailoring files leaves the administration to the individuals and bears the danger of having numerous and possibly inconsistent local copies of tailoring files. However, it is feasible where someone wants to share a tailoring file with a particular other person or group. Again in analogy to the treatment of other files, it is recommendable to provide support for both forms of sharing tailoring files and support the users to let effective and efficient ways of usage emerge.

Depending on the technical and organizational structure, also hybrid forms may emerge: peer-to-peer exchange of tailoring files may be enhanced to sending a tailoring file to a whole group that the sender defines or to a list that people can subscribe to e.g. if they want to receive all tailoring files from a particular author. Thus, a mailing mechanism turns into a broadcast mechanism bearing similarity to a shared workspace where the entries are divided into subfolders depending on the author of a tailoring file.

The literature on collaborative tailoring and my own work has dealt with small groups of no more than 10 people. It seems, that this is a reasonable group size to share tailoring experiences and files since people in a group of this size may also share tasks or work in very similar settings which makes collaborative tailoring in a stricter sense reasonable. For this group size, a shared workspace with some structure and access rights and some basic mailing mechanisms should be enough to support collaborative tailoring. For much larger groups other mechanisms like newsgroup-like broadcasting of tailoring files might be more useful. However, it remains open if such a large group is likely to have similar tailoring interests and how *collaboration* in such a group could be reasonably defined.

In the case of the word processor several mechanisms for sharing, sending and receiving tailoring files were provided: a shared workspace to provide a location to store tailoring files; mailing mechanisms for users to be able to send tailoring files directly to other single users and groups of users; and a private workspace for tailoring files that may be copies of files from the public store or files received from others via the mailing mechanism. The laboratory test showed that this distinction was understood (see Kahler 2000). A prototype of the groupware search tool allowed users to share tailoring files by saving them in a shared workspace. There, they could be deleted, renamed or copied (see Kahler et al. 2001).

Also in the literature several authors report on different forms of sharing tailoring files (see also section *Related Work*). Mackay (1990) reports on various forms and preconditions of sharing tailoring files ranging from telling someone how to tailor to reach a certain effect to actively putting a tailoring file into a predefined shared workspace. MacLean et al. (1990) explicitly support sending tailoring files to others with their Buttons system. Wasserschaff & Bentley provide mechanisms that their Tviews (tailoring files describing a particular view on a document) can be distributed via the groupware system that they are programmed for. In the discussion about

collaborative tailoring sharing tailoring files is considered a necessity to support collaboration. Henderson & Kyng (1991, p. 233) state that “*means must be available to acquire changes*”; Bentley & Dourish (1995, p. 145) require that it be “*possible to add attachments to the shared workspace for others to retrieve and use*”.

Suggestion 3: Allow Browsing Through Tailoring Files

Browsing helps people to find information they need even if they may not look for a specific information. Particularly for tailoring files it provides support on two levels. Firstly, users browsing through tailoring files may find files that they are interested in with functions that they had not thought of before. Secondly, on a higher level browsing through tailoring files is a means to get an overview over the tailoring of a group particularly for new group members. By browsing they can get an impression of the number of files, the structure of folders possibly representing different areas of tailoring, the annotations and automatic descriptions provide information about active tailors and their tailoring focus. In the word processor case the possibility to browse through tailoring files including the display of annotations was provided (see suggestion 5: *Make Annotations and Automatic Descriptions Possible*). Thus, the users could get a first impression of the tailoring file supporting their decision-making whether the tailoring file is interesting for them or not. More advanced features here could include sub-structuring folders of tailoring files, listing several attributes of tailoring files (e.g. name, creator, date of creation, size, number of people who downloaded or saved it from the shared workspace, or an average of all marks given to that tailoring file by all users who tried it out) and then making it possible to sort the tailoring files by their attributes so a list of tailoring file results sorted by the creator or by the quality as judged by other users.

Catledge & Pitkow (1994) distinguish three forms of browsing, namely search browsing (i.e. directed search) where the goal is known, general purpose browsing as consulting sources that have a high likelihood of items of interest and serendipitous browsing which is purely random. They claim that browsing is adequate for open tasks that not have a specific answer and are more subject oriented. Twidale et al. (1997) concede that when users seek information their needs are often initially vague and evolve during the search process, so that general purpose browsing is a more accurate

description of users' behavior than searching. According to Chang & Rice (1993) general browsing has been construed as a search strategy, a viewing pattern, a screening technique, and a recreational activity. They argue that in connection with directed search there are some unrealistic assumptions about users and the nature of information seeking - e.g. that users have unbounded rationality, have static and well-defined information needs, know what they want and are output oriented. Rather, they claim, users often do not have predefined search criteria, and may alter their interests during a search.

The interviews in the word processor case and the experience in the groupware search tool case revealed that users are willing and interested to use others' tailoring files but they often do not know if there exists a tailoring file that may meet their needs. This is in line with the observations of Mackay (1990) on patterns of sharing tailoring files where several of the methods to obtain or give tailoring files support the notion of vague initial needs satisfied by asking others and stepwise finding out what it really is that you want to know. Consequently, Mackay concludes that tailorable software should provide the ability to browse through others' useful ideas.

Technically browsing can be supported by adequate browsing facilities that can give both a general impression about what files are available and, in connection with other features like annotations (see suggestion 5), can provide relevant information e.g. about the author or a brief description of the functionality at a quick glance. Also, different sorting mechanisms for tailoring files in a directory can provide an overview and helpful impression e.g. about the number of tailoring files provided by an author, the age of tailoring files or even on the content if some keywords are provided. Bearing in mind, that collaborative tailoring is currently rather an activity of small groups lightweight solutions seem to be preferable over very elaborate browsing tools that may incorporate advanced database technology but are difficult to handle and require much input by the creator of the tailoring file.

Interestingly enough, the concept of general browsing has had an enormous upswing with the growth of the World-Wide Web. On one hand this is obvious considering the hyperlink structure of the WWW. However, the sheer mass of files in the WWW seems to make finding strategies more adequate that contain strong aspects of direct logical or keyword search with a search engine, whereas general browsing seems a more suitable concept

for small or medium sized collections of files that may be pre-structured by a folder structure or keywords or any form of subject trees.

Suggestion 4: Provide Awareness of Tailoring Activities

Collaboration needs information about what others do. This also concerns the need for awareness about others' tailoring. Henderson & Kyng (1991, p. 233) explicitly relate to collaborative tailoring when they claim that "*news must be published that change is available*". This helps users to stay current and avoids double work on the same tailoring issue.

Several interviewees in the word processor case reported the importance and different forms of making others aware of their tailoring work. Interestingly enough, the interviewees also report non-technical awareness mechanisms. One of these consists in putting a notice on a non-electronic blackboard. Another of these mechanisms is just telling a friend that they tailored something that might be worthwhile for the other person. Mackay (1990) reports on similar experiences and lists several methods to obtain or give tailoring ideas ranging from someone else telling a person about a new tailoring idea or file to electronic postings of new ideas in a common area.

The term and notion of *awareness* have been very popular in CSCW research in the last couple of years since awareness is considered to capture many aspects of what makes collaboration successful. The basic definition is provided by Dourish & Bellotti (1992, p. 107) who define awareness as being "*an understanding of the activities of others which provides a context for your own activity*". Gutwin (1997) distinguishes several forms of awareness of others in collaboration: *Informal Awareness*, the general sense of who is around and what they are up to, *Conversational Awareness* consisting of visual and verbal cues providing a sense of what is happening in a conversation, *Structural Awareness* involving the knowledge of a group's organization and the working relationship and *Workspace Awareness* being the up-to-the-moment understanding of another person's interaction with a shared workspace. All of these forms can play a role for collaborative tailoring. Structural and informal awareness are important since they may support the notion of the existence of a tailoring culture (see suggestion 8: *Support a Tailoring Culture*). These forms of awareness may not only generally encourage tailoring and collaborative tailoring activities but also support the users' knowledge that they can share tailoring files or that somewhere in the system there may be a place for tailoring files that

they can look at when they want to browse and that they can put their tailoring files.

All aspects of awareness in the general context of collaboration can be applied for collaboration in the context of tailoring. In the latter case, awareness gains a particular importance since tailoring for most people is not their primary work task. Therefore, they might not actively be concerned about informing themselves about others' tailoring and need particular mechanisms to be kept informed.

One way to create workspace awareness is by notifying others of the existence of tailoring activities or tailoring files. If tailoring files are shared this can be accomplished by a simple event driven notification service stating that a person received a tailoring file from someone else or that someone has uploaded a new tailoring file in the shared workspace.

In word processor case mentioned above several forms of awareness play a role: the awareness about others' tailoring is provided by a notification service that is triggered whenever a tailoring file arrives in the tailoring inbox. The notification service informs the user via message window at start up time of the word processor and at the time the user activates a tailoring function in the menu. In a workshop the time of notification was a topic of discussion since all users wanted some notification but some were afraid to be "overnotified" and bothered too much by what they considered relevant but not urgent information. It is one of the practical challenges to find an adequate awareness mechanism for each setting, person, group, time, or tailoring file or activity. The notification window of the word processor extension presents the tailoring files and asks the user either to store it in his or her private repository or to delete it instantly. Another simple form of awareness is provided by the fact that the browser for tailoring files always shows the list of tailoring files in the shared workspace so that new files can be recognized. This direct form of letting others know that someone is engaged in tailoring is only one of many ways that awareness can support collaboration by. Other forms of workspace awareness for collaborative tailoring may include the generation of automatic mails, a ticker tape with current information about tailoring activities like the number of tailoring files uploaded to the shared workspace in that week or list of tailoring files or activities that can be easily accessed and may include particular markers for list items that have been changed or added since the list was last viewed. More advanced mechanisms include the definition of interest context that

someone can subscribe to and is informed when something in this context changes where the context can be determined automatically by the system or be put in manually by the creator of the tailoring file. Note, that annotations (see below) can be considered to provide awareness by making context of the tailoring explicit.

Suggestion 5: Make Annotations and Automatic Descriptions Possible

For collaboration it is helpful to understand the context in which the other participants work. As described above, creating awareness is one means to provide understanding of context. In the groupware search tool case it became clear after a while that it was difficult for others to understand what a particular search tool created by one person did. Therefore the participants required to provide more context by a textual description. Such annotations serve to enhance learning and share that understanding with others (Henry 1997). Active annotations, i. e. adding a critical or explanatory note to a file, are a good means of providing context particularly to a shared file. This is more necessary for a tailoring file than for a general file concerning the primary work task since the tailoring file is usually less self-explanatory and has a form that is less known to users than the files they usually work with. Here, annotations can serve to explain the context and the function of the tailoring file. Similar to remarks added by programmers to the program code they write annotations added to tailoring files by tailors serve to provide a better understanding of what the tailoring file's function is. The annotations to a tailoring file should be distributed with the tailoring file and be should be visible without having to open or use the tailoring file. Annotations can be made as plain text provided by the tailor to go with the tailoring file. The system can also provide automatically generated descriptions of parts of the context that the tailoring file was produced in.

One difficulty with annotations as compared to automatic descriptions may be that there must be time spent to write the annotation. As with all documenting the time and effort to describe what someone did should be in a reasonable ratio to the time the actual work (here: tailoring) was done. Considering the fact that tailoring itself is usually not the primary task for the person who tailors a basic willingness can be assumed to spend time on tailoring and a little effort on annotating the tailoring file which may be the decisive difference that makes others use the tailoring files.

The advantage of automatic descriptions is that there is no additional work on the side of the tailor needed. Several information already available in the system can be used for automatic description, e. g.:

- the provision of the author of a tailoring file can be useful to contact her to understand something better or to complain. If tailoring is an accepted activity it is beneficial for the author to be credited the tailoring file;
- the automatic provision of the version of the tailoring file or the name of the file that it was derived from provides is an important information for people who like the tailoring file and are keen on a debugged or extended version;
- technical information on what is contained in the tailoring file (e.g. macros and toolbars contained in a word processor's document template) provides additional structure.

Which of this information is really useful to go with the tailoring file can depend on different aspects like the application, the part of it which is tailored or the user viewing the automatic description. Good default settings are particularly valuable here and should be subject to a discussion within a group sharing tailoring files. Moreover, the automatic descriptions cannot be expected to provide as rich an information as the annotations that may provide context not only by listing statuses but by letting the annotation author draw context links between these and describe motivation for or embedding of the tailoring file.

Depending on how refined a substantial percentage of the annotations are expected to be a further subdivision into several topic-related plain text fields for annotation or the assignment of keywords to create some additional structure can be useful. This could support browsing through tailoring files (see suggestion 3) which could be presented or sorted by keywords or the content in certain plain text annotation fields. While on one hand it may be nice for the author of an annotation to be free what to write in a plain text annotation field it seems to be recommendable that conventions about the usage of the plain text field are supported so that the reader understands if the author intends to describe the motivation of the tailoring activity or the technical aspects of the tailoring file or the task that the tailoring file supports.

The annotations can be considered to be a communication channel between the persons writing and reading them. Moreover, both annotations and automatic descriptions may serve to make exploration easier since they already provide information about the functionality of a tailoring file.

In both the word processor and in the search tool case the possibility for textual annotation of tailoring files was provided. In each case the annotation information could be seen when browsing through the list of tailoring files. In the word processor case the annotation information could also be seen when receiving a tailoring file from another person by means of the mailing mechanism. In this case, it supported the decision if the received file should be opened or saved or if it should rather be deleted.

Mørch (1995b) provides context in a similar way by a presentation of the *rationale* in his layered architecture for tailorable applications.

Suggestion 6: Allow for Exploration of Tailoring Files

Tailoring files may include aspects of presentation, e.g. a corporate letterhead, of manipulation, e.g. a toolbar for special tasks, or of action, e.g. in a macro. Even with context provided e.g. by annotations it is not always clear to other users what effects a tailoring file has, particularly if it includes actions or a set of different tailoring aspects. One way to support understanding here is to provide means for *exploration*, i.e. finding out what a tailoring file “does” without necessarily producing all the effects of the tailoring file persistently, thus avoiding the danger of deleting previous work by some unforeseen effect of somebody else’s tailoring file.

Note the difference of browsing and exploration: While *browsing* through tailoring files means to pass by a collection of tailoring files and by this get an overview over tailoring files and activities, *exploring* a tailoring file means to learn more about a tailoring file without having to fear that existing data are deleted. This is particularly important for more complex tailoring files which themselves do not only present but also manipulate data. This is the case e.g. for a macro contained in a word processor document template, which may format a document in a certain way. Exploration can be considered to be a form of self explanation of the system.

According to Engelskirchen (2000) *investigative* and *experimental* exploration can be distinguished. The former is an excursion through the system where the user takes a look at the self-descriptions offered by the

system. By the latter the user actively builds and tests hypotheses by just trying something out and looking what happens. Also exploration can be strongly interwoven with use of a system when a new use situation arises, a user tries something out and afterwards may use the knowledge gained to continue working.

Engelskirchen names several mechanisms which can support exploring particularly in collaborative settings:

- *Help texts and tutorials*: help texts and tutorials can provide static and basic information about a file or system. However, in dynamic settings, these static information should be enhanced by dynamic information e.g. by annotation to the help texts.
- *Exploration cards*: exploration cards are more task-oriented and point to particular relevant information. They are basically note cards with entries (steps of task, success control, corrective steps) serving to guide the user through some particular task. Exploration tasks are less static than help texts and tutorials and can serve to guide a user through a particular task also in a non-default setting.
- *Interaction histories, graphs and filters*: Interaction histories list actions performed by a user and help a user to understand how she got to a certain point in using a system. In a collaborative setting, however, the state of the system may depend on several users so a single person's interaction history may not suffice to understand the way to the system's current status. Interaction histories may be visualized by *interaction graphs*. Several forms of *filters* serve to adequately reduce an interaction graph so a user can more easily detect relevant aspects of his actions that led him where he is now.
- *Cancellation*: Cancellation serves to undo a user's action. The possibility of cancellation can provide users with a feeling that they can try out a system without damaging anything. In collaborative settings cancellation is often difficult due to the fact that it would include undoing other users' actions.
- *Neutral mode*: In the neutral mode an activated function is not carried out, but a text explains what would have happened if the function had been carried out. Therefore, in the neutral mode no damage can be done to the system, but also the result of an action is not really experienced.

- *Freezing points*: Sometimes it can be helpful for a user to define a system status, the freezing point, that she wants to return to before she starts trying out the system's functions.

Note that annotations (see suggestion 5) could be considered to be means of exploration, too, since they, like help texts for a general, provide information on a tailoring file. However, in this context they are not primarily considered to be formal means of information but more to be means of more or less formal communication from the creator of the tailoring file to its user.

In the word processor case a preview mode served to support investigative exploration since with the preview mode could get a first impression what a document template looked like and decide if they wanted to save it at all.

In the search tool case exploration was a critical issue, since using and trying out the search tool involved the actions and settings of many other users and thus became a rather complex affair. Particularly for the user acting as local expert being able to test newly created search tools was critical since he wanted to know whether the tailored search tools did what they were supposed to do.

Suggestion 7: Make Administration and Coordination Easy

Means should be provided to administer and coordinate the tailoring activities of a group. Administration of tailoring files relates to the static aspects of keeping existing tailoring files in a way that they are useful for a group whereas the coordination of tailoring files and activities relates stronger to dynamic and creative aspects of tailoring where the aim is to possibly create synergetic effects.

The interviews of the word processor case revealed the importance of such administration and coordination for several interviewees e.g. in relation to organization-wide document templates for administrative purposes. In one of the use situations the members of the organization find document templates of administrative purpose on one of these intranet servers (e.g., ordering and billing forms). These templates are created and updated by a central organizational unit. All the other users can just copy these templates and ideas for new forms have to be proposed to that unit. In another use situation in the public administration strong aspects of coordination could be

found. One of the employees from the administration site reported how she created a document template together with a colleague. Both of them carried out parts of the whole job. Then she put her part of the template on a disk and carried it to her colleague who pasted the parts together. In yet another use situation both administrative and coordinate aspects were mixed: everybody in that department had used her or his own mode to create particular tables. The interviewee started to standardize the layout of these tables by creating a first template containing some macros. He then discussed it with his colleagues. Having found an agreement with them, he asked his boss for a final approval. Thus, the interviewee bundled and coordinated the activities of several people. Then, an administrative effort followed: he put the templates on the LAN giving most of the persons of his department read and write permission. One of the users of whom he thought that he would endanger the template due to lacking skills was just granted read permission but no permission to write. When everything was set up, the interviewee informed his colleagues verbally about the location of the shared template on the LAN.

The administration of tailoring files is necessary particularly with a growing number of tailors and tailoring files. The administrator is a person to have an overview over tailoring, thus being able to order the tailoring files. Administration of tailoring files resembles the administration of an operating system or data base in general. The administration may include taking care of a shared workspace by removing old versions or by checking and debugging files that are put into the shared workspace. The administrator may also to compare and combine several tailoring files. This requires administrative access rights and possibly the installation of a tailoring sandbox where a tailoring file can be tested (see also suggestion 6: *Allow for Exploration of Tailoring Files*). In the word processor case administrative rights were provided to delete files from the shared repository.

Henderson & Kyng (1991) suggest that systems used together demand that tailoring should be coordinated. The coordination of tailoring may involve a cross-section of technical mechanisms and organizational measures to support efficient collaboration. While an administrator focuses on work with tailoring files, the coordinator (which may be the same person) focuses on the collaboration of different users and groups. Tailoring files can be considered to be artifacts around which collaboration evolves and subsequently coordination is necessary. The coordination may include

eliciting and realizing technical and organizational requirements to support collaborative tailoring. Also, an active information policy about tailoring e.g. by email, or the organization of workshops or a mentor system concerning collaborative tailoring can play an important role. Particularly a person coordinating tailoring activities of a group has a critical position to foster the tailoring culture (see suggestion 8). She must invent and apply technical mechanisms for coordination and have a standing within the group in order to motivate them to tailor and make tailoring a collaborative effort even if that may take time off their primary work task.

Suggestion 8: Support a Tailoring Culture

Besides and in addition to technical measures the success of collaborative tailoring also depends on socio-organizational aspects like a *tailoring culture*. The necessity to understand that tailoring and particularly collaborative tailoring can bring benefits to a group working together or to an organization was a prerequisite for our application partner in the search tool case (see Kahler et al. 2001) to identify one colleague to be responsible for coordinating and administering collaborative tailoring. The interviews in the word processor case revealed several forms of an emerged or installed tailoring culture ranging from a person-to-person exchange of tailoring files or ideas to persons acting as local experts, some of them being officially recognized. The interviews also show that tailoring culture is often developed in a bottom-up process where few persons tailor, then exchange ideas and files, a knowledgeable and interested person emerges as local expert and ideally these efforts are institutionalized by officially recognizing the tailoring activities or nominating someone officially responsible for coordinating and administering collaborative tailoring.

In their Buttons system MacLean et al. (1990) explicitly supported the sending of tailored files via email. The users in their study in the beginning had mixed feelings towards the buttons and perceived them as unfamiliar and messing up the screen. However, after a while they got used to the buttons and did not only share buttons with others but also over time appropriated buttons and started to perceive them as personal and positive. The notion of the importance of a community of people who tailor is supported by Carter & Henderson (1990). Based on their experiences with the Buttons system they claim that a tailoring culture is essential to the effective use of a tailorable technology. Such a tailoring culture grows as

tailoring becomes part of users' everyday work and makes them experience the technology as being under their control. They conclude that (p. 113)

“tailorability is a relationship to rather than a property of technology. Tailorability addresses how technology fits into an organisation and how groups and individuals make use of it.”

The steps of appropriating tailoring and tailoring files in everyday work and thus creating a tailoring culture is represented by three possibly coexisting levels of tailoring culture:

- **level of equals:** An important part of a tailoring culture is that people help each other and collaborate in tailoring who work on the same level and possibly on similar tasks. They are the ones who know best what kind of a tailoring file may be suitable for their current work or for a colleague. In our word processor case we found the case of two persons collaboratively providing a document template that was used by their whole work group and a use situation where two law students exchanged useful tailoring information. Gantt & Nardi (1992) report of CAD (Computer Aided Design) users cooperating in tailoring their environments or programming applications, and Trigg & Bødker (1994) even found a “network of who-asks-whom” (p. 51), where often the nearest co-worker is the first person to ask.;
- **different levels of expertise:** For more complex tailoring tasks it is helpful to be able to have a more experienced user around who has a close contact with the people of a particular work group. Often, this is a member of that group whose technical interest or just some chance of being in need and able to learn how to tailor makes her a local expert. Carter & Henderson (1990) use the term *transactor* for a person who mediated between users, designers, and technology and whose role emerged to be central to the development of a tailoring culture. Gantt & Nardi (1992) identify *gardeners* and *gurus*, who help end users write, complete and debug CAD macros and themselves write macros and shell scripts that are beyond the scope of end users' abilities;
- **organizational embedment with tailoring as a community effort:** The development and preservation of a tailoring culture benefits enormously from an organizational embedment and recognition of tailoring activities. This prevents tailors from work overload from tailoring being an unrewarded additional activity and may create additional value for

the organisation. In the case of Gantt & Nardi's (1992) CAD tailoring, the local developers were given recognition, time and resources for pursuing their activities while their managers benefit from "someone who can be officially be relied upon to help end users, and to maintain standardization of the macros and programs they use" (p. 112). The official recognition and organizational embedment of tailoring should be part of and go hand in hand with tailoring being a community effort (see MacLean et al. 1990).

The efforts for establishing and maintaining a tailoring culture must certainly be in line with those for a general organizational culture but with the advantage of a smaller and thus better manageable scale and more concrete options for action.

Conclusion

The proposed suggestions for collaborative tailoring concern technical and socio-technical aspects of system design and use. Similar observations and recommendations have been made in other sub-disciplines of information science, computer science or organizational science, so none of the single suggestions is new in itself. However, they are put together here with the particular focus on collaborative tailoring. On one hand, this focus allowed a careful selection based on previous empirical work particular on tailoring and collaborative tailoring. On the other hand the focus allows a level of concreteness that clearly goes beyond just general statements about how software and its use in organizations should be designed. e.g., while the notion of organizational culture usually stays fuzzy, the notion of tailoring culture, which can be considered part of organizational culture, is much more clear-cut and can be related to measurable indicators such as official recognition of tailoring activities.

The eight suggestions stem from several backgrounds from file management via CSCW to the organizational aspects of information systems and cover overlapping areas of tailoring and complement each other. Depending on the organizational setting, work task or refinement of the system used they may be of different importance. There are cases where it may be most important to have different ways to share tailoring files while in other cases one simple mechanism suffices. In again other settings exploration of tailoring files may be completely unnecessary if they are well annotated whereas in other

cases insecurity prevails about what exactly the files does and if the tailor has implemented everything correctly. Thus, the suggestions can be considered to be concrete enough to put into practice but still leave room for adjustment to a particular setting. This usually will bring “hard” technical and “soft” human and organizational aspects of the suggestions to go hand in hand.

One of the limits of collaborative tailoring is certainly drawn by the answer to the question how efficient it is. However, this may be as difficult to measure as the efficiency of working with groupware or a common file system or the answer to the question how useful are standards. While no single answer may be reasonably provided for this, the literature review and my own experience clearly shows that there are positive cases in which technical and organizational difficulties to collaborative tailoring were overcome in order to establish a working and beneficial culture of collaborative tailoring.

References

- Bentley, Richard; Dourish, Paul (1995): *Medium vs. Mechanism: Supporting Collaboration Through Customization*. In: Proceedings of ECSCW '95. Kluwer. pp. 133-148.
- Carter, Kathleen; Henderson, Austin (1990): *Tailoring Culture*. In: Reports on Computer Science and Mathematics no. 107, Åbo Akademi University 1990. Proceedings of 13th Information Systems Research Seminar in Scandinavia (IRIS). pp. 103-116.
- Catledge, Lara D.; Pitkow, James E. (1995): *Characterizing browsing strategies in the world wide web*. In: Computer Networks and ISDN Systems, Vol. 27 (6). pp. 1065-1073.
- Chang, Shan-ju; Rice, Ronald E. (1993): *Browsing - a multidimensional framework*. In: Annual Review of Information Science and Technology, Vol. 28. pp. 231-276.
- Dourish, Paul; Bellotti, Victoria (1992): *Awareness and Coordination in Shared Workspaces*. In: Proceedings of CSCW '92. pp. 107-114.
- Engelskirchen, Torsten (2000): *Explorationsunterstützung in Groupware am Beispiel eines anpaßbaren Suchwerkzeugs (English: Supporting Exploration in Groupware: the Example of a Tailorable Search Tool)*.

Department of Computer Science III, University of Bonn. Diploma Thesis.

Gantt, Michelle; Nardi, Bonnie A. (1992): *Gardeners and Gurus: Patterns of Cooperation among CAD Users*. In: Proceedings of CHI '92. pp. 107-117.

Grudin, Jonathan (1991): *Interactive Systems: Bridging the Gaps Between Developers and Users*. In: IEEE Computer, April 1991. pp. 59-69.

Gutwin, Carl (1997): *Workspace Awareness in Real-Time Distributed Groupware*. Department of Computer Science. Calgary, Alberta, University of Calgary. Ph.D. Thesis.

Haaks, Detlef (1992): *Anpaßbare Informationssysteme - Auf dem Weg zu aufgaben- und benutzerorientierter Systemgestaltung und Funktionalität*. Göttingen und Stuttgart, Verlag für Angewandte Psychologie.

Henderson, Austin (1997): Tailoring Mechanisms in Three Research Technologies. In: Workshop on Tailorable Groupware: Issues, Methods, and Architectures at the Group '97 organized by Mørch, Anders; Stiernerling, Oliver; Wulf, Volker. <http://www.ifi.uib.no/staff/anders/research/group97/wspapers/henderson.ps>.

Henderson, Austin; Kyng, Morten (1991): *There's No Place Like Home: Continuing Design in Use*. In Greenbaum, Joan; Kyng, Morten: Design at Work - Cooperative Design of Computer Systems. Hillsdale, NJ, Lawrence Erlbaum Associates. pp. 219-240.

Henry, Paul David (1997): *PALIMPSEST - a model for Networked Hypermedia and Distance Learning*. <http://www.programhouse.com/pal/paltext.htm>.

JCSCW (2000): *JCSCW - Computer Supported Cooperative Work (CSCW): The Journal of Collaborative Computing*. Vol. 9 (1). Special Issue on Tailorable Systems and Cooperative Work.

Kahler, Helge (2000): *Constructive Interaction and Collaborative Work: Introducing a Method for Testing Collaborative Systems*. In: acm interactions, Vol. VII (3 (May/June 2000)). pp. 27-34.

Kahler, Helge (2001): *More Than WORDs - Collaborative Tailoring of a Word Processor*. In: Journal of Universal Computer Science (j.ucs), accepted for publication.

- Kahler, Helge; Stiernerling, Oliver; Won, Markus; Wulf, Volker (2001): *Tailoring by Integration of Components: The Case of a Document Search Tool*. submitted for publication.
- Mackay, Wendy E. (1990): *Patterns of Sharing Customizable Software*. In: Proceedings of CSCW '90. pp. 209-221.
- Mackay, Wendy E. (1991): *Triggers and Barriers to Customizing Software*. In: Proceedings of CHI '91. pp. 153-160.
- MacLean, Allan; Carter, Kathleen; Lövstrand, Lennart; Moran, Thomas (1990): *User-Tailorable Systems: Pressing the Issues with Buttons*. In: Proceedings of CHI 90. pp. 175-182.
- Malone, Thomas W.; Grant, Kenneth R.; Lai, Kum-Yew; Rao, Ramana; Rosenblitt, David (1988): *Semistructured Messages are Surprisingly Useful for Computer-Supported Coordination*. In: Proceedings of CSCW 88. Morgan-Kaufmann Publishers. pp. 311-334.
- Mørch, Anders (1995a): *Three Levels of End-user Tailoring: Customization, Integration, and Extension*. In: Computers in Context: Joining Forces in Design. Aarhus. pp. 157-166.
- Mørch, Anders (1995b): *Application Units: Basic Building Blocks of Tailorable Applications*. In: Proceedings of East-West Int'l. Conference on HCI. Springer. pp. 68-87.
- Nardi, Bonnie A.; Miller, James R. (1991): *Twinkling lights and nested loops: distributed problem solving and spreadsheet development*. In: Int. J. Man-Machine Studies, Vol. 34. pp. 161-184.
- Oppermann, Reinhard, Ed. (1994): *Adaptive User Support: Ergonomic Design of Manually and Automatically Adaptable Software*. Hillsdale, New Jersey, Lawrence Erlbaum Associates.
- Stallman, Richard (1981): *EMACS: The Extensible, Customizable, Self-Documenting Display Editor*. In: Proceedings of ACM SIGPLAN SIGOA. Massachusetts Institute of Technology. pp. 301-323.
- Trigg, Randall; Bødker, Susanne (1994): *From Implementation to Design: Tailoring and the Emergence of Systematization in CSCW*. In: Proceedings of CSCW '94. pp. 45-54.
- Trigg, Randall H. (1992): *Participatory Design meets the MOP. Accountability in the design of tailorable computer systems*. In:

Proceedings of 15th Information Systems Research Seminar in Scandinavia (IRIS). pp. 643-656.

Trigg, Randall H.; Moran, Thomas; Halasz, Frank (1987): *Adaptability and Tailorability in NoteCards*. In: Proceedings of Human-Computer Interaction - Interact'87. pp. 723-728.

Twidale, Michael B.; Nichols, David M.; Paice, Chris D. (1997): *Browsing is a Collaborative Process*. In: Information Processing & Management, Vol. 6 (33). pp. 761-783.

Wasserschaff, Markus; Bentley, Richard (1997): *Supporting Cooperation through Customisation: The Tviews Approach*. In: Computer Supported Cooperative Work: The Journal of Collaborative Computing (JCSCW), Vol. 6 (4). pp. 305-325.