In: Proceedings of Designing Interactive Systems (DIS) ´97, Amsterdam 18.-20. August 1997, pp. 365 - 376

How to Make Software Softer - Designing Tailorable Applications

Oliver Stiemerling, Helge Kahler, Volker Wulf

University of Bonn
Institute for Computer Science III
Römerstr. 164
53117 Bonn
{os, kahler, volker}@cs.uni-bonn.de

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.

KEYWORDS
Tailorability, groupware, participatory design, design cases

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 ([29], [4], [18], [11], [25], [19], [14], [1]).

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. [11], [10], [21] stress the basic flexibility of object oriented architectures in this respect. For instance, OVAL (see [19]) 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 [5]) 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 divers 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 [22], [11]). 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 [17]) 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 [2]) 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.

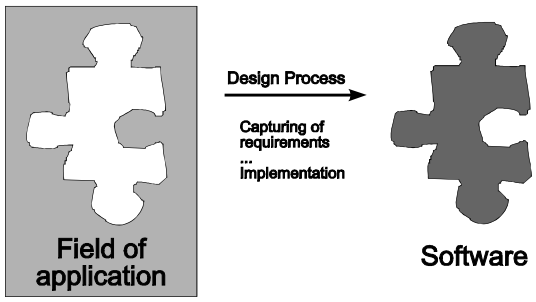

Fig. 1: Designing software which meets the requirements of one field of application at one time

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 [3], [12]). 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 [7], [8]).

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, [31] 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. ([7]).

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

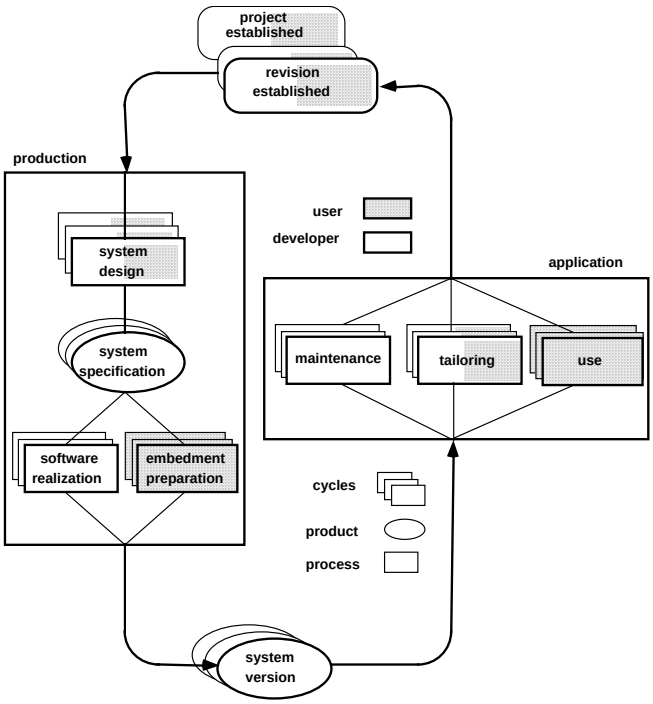

Fig. 2: Extended STEPS-approach - Combining evolutionary and participative software development with tailoring in use (see [31])

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 [11], [30], [26], [24]). 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.

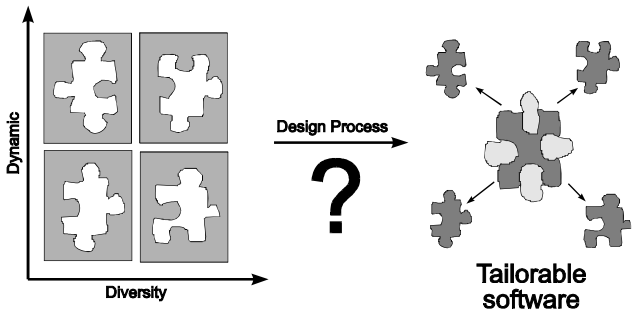

Fig. 3: The central question of this paper: how to design tailorable software?

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 [16] have applied participatory techniques in analyzing the requirements for flexibility of a picture archive and communication system in a hospital. They conclude "... *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*" (p. 22).

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 in [9]. 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 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. ([6]) suggest to extend Jacobsen´s *use-case* methodology (see [13]) 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:

- "*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*" (p. 354)

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 [20]). 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.

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:

1. foremost the end users implement access policies, not the system administrator.
2. 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.
3. 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

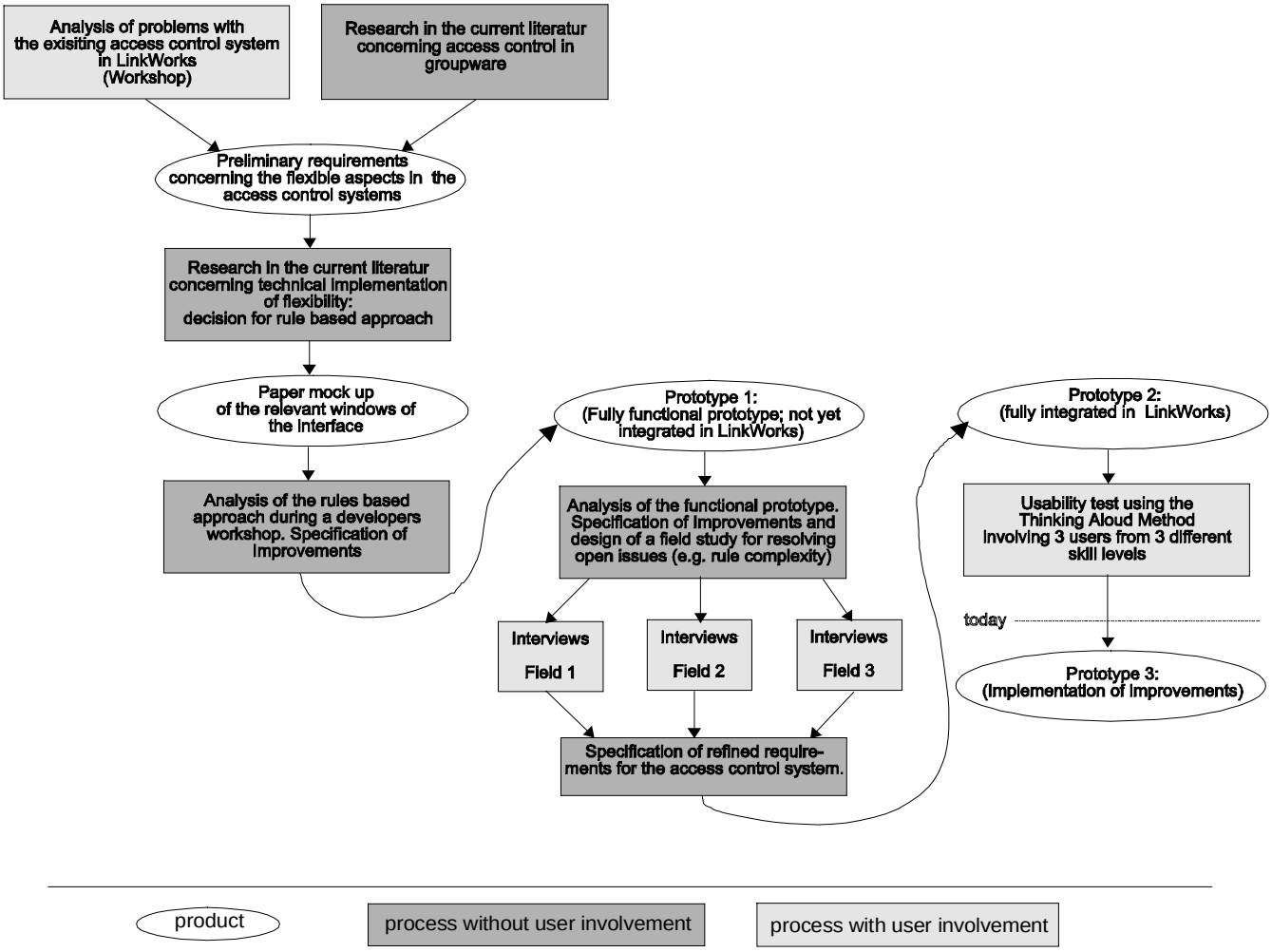

Figure 5: Design process for the new access control system

"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:

organization / *person*	public	semi-private	private
superior			
subordinate			

Fig. 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?").

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 ([28], [27]) 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 show 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.

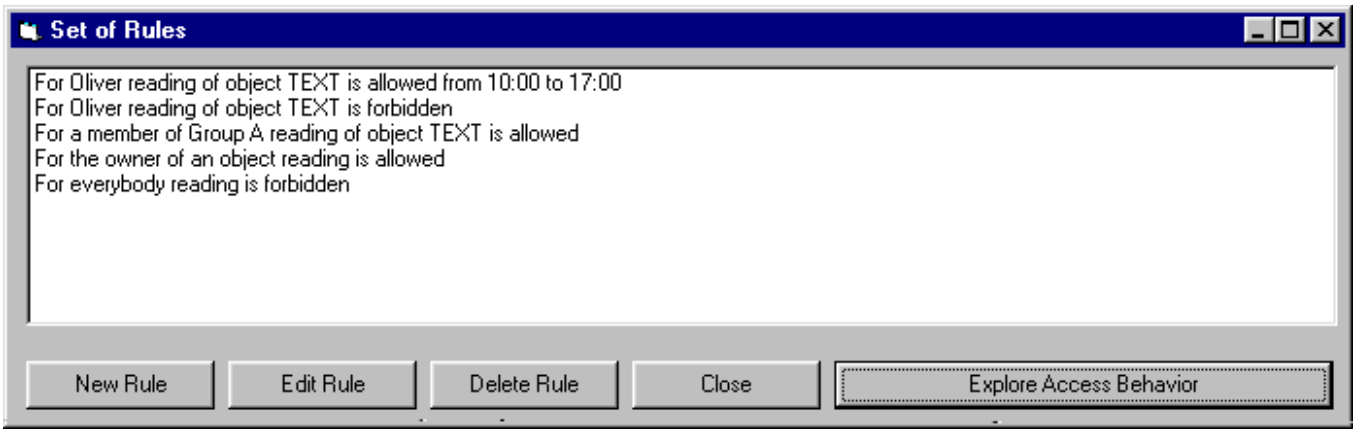

Figure 7: Screen presenting access rules to the end user

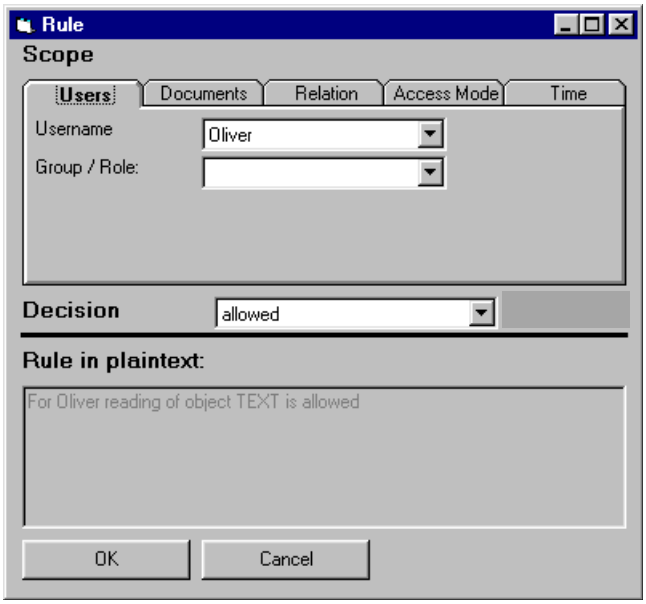

Fig. 8: Screen for editing a single rule

We used the Thinking Aloud Method (see [23]) 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 polices 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 [15].

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 fig. 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 [20]) helped us to get a better understanding of the work of our application partners and their requirements.

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

Figure 9: Redesign process for the search tool

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 fig. 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.

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

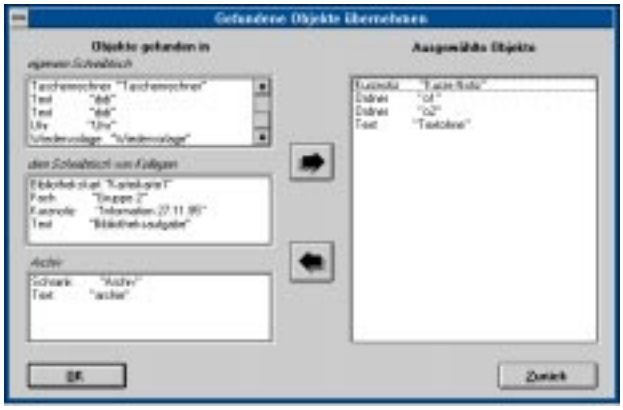

Fig. 10: Output dialog of search tool showing where objects were found

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 reimplementation 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

1. Bentley, R. and Dourish, P.: Medium versus Mechanism. Supporting Collaboration Through Customisation, in: Marmolin, H.; Sundblad, Y.; Schmidt, K. (Hrsg.), Proceedings of the Fourth European Conference on Computer Supported Cooperative Work - ECSCW '95, Kluwer, pp. 133-148.
2. Boehm, B. W.: Software Engineering, in: IEEE Transactions on Computers, C-25 (1976), 12, pp. 1216-1241.
3. Boehm, B. W.: A Spiral Model of software development and enhancement, in: Computer, 5/1988, pp. 61-72.
4. Carter, K., and Henderson, A.: Tailoring Culture, in: Hellman, R., Ruohonen, M., Sørgaard, P. (eds.): Proceedings of the 13th IRIS. Reports on Computer Science and Mathematics no. 107, Åbo Akademi University 1990, pp. 103-116.
5. DEC, Digital Equipment Corporation: LinkWorks Version 3.0 - User's Guide. Part number AA-Q3KYB-TE, 1995.
6. Ecklund, E.F.; Delcambre, L.M.L.; Freiling, M.J.: Change Cases: Use Cases that Identify Future Requirements, in: Proceedings of OOPSLA '96, CA, USA, ACM Press, 1996, pp. 342-358.
7. Floyd, Ch; Reisin, F.-M.; Schmidt, G.: STEPS to software development with users, in: Ghezzi, C.; McDermid, J.A. (eds.): ESEC'89 - 2nd European Software Engineering Conference, University of Warwick, Coventry, Lecture Notes in Computer Science No. 387, Heidelberg, Springer, 1989, pp. 48 - 64.
8. Grønbæk, K., Kyng, M., Mogensen, P.: Cooperative Experimental System Development - Cooperative Techniques Beyond Initial Design and Analysis, in: Proceedings of the 3rd Decennial Conference: Computers in Context: Joining Forces in Design, Aarhus, Denmark, August 14-18, 1995, pp. 20-29.
9. Grudin, J.: Why groupware applications fail: problems in design and evaluation, in: Office: Technology and People, 4. Jg., No. 3, 1989, pp. 245-264.
10. Haaks, D.: Anpaßbare Informationssysteme - Auf dem Weg zu aufgaben- und benutzerorientierter Systemgestaltung und Funktionalität, Göttingen u.a. 1992.
11. Henderson, A. and Kyng M.: There's No Place Like Home. Continuing Design in Use, in: Design at Work, Lawrence Erlbaum Associates, Publishers, 1991, pp. 219-240.
12. Henderson-Sellers, B. and Edwards, J.M.: The object-oriented System Life Cycle. In: Communications of the ACM, Vol. 33, 9, pp. 143-159.
13. Jacobsen, I.; Christerson, M.; Jonsson P.; Övergaard, G.: Object-Oriented Software Engineering: A Use Case Driven Approach, ACM Press, 1992.
14. Kahler, H.: From Taylorism to Tailorability: Supporting Organizations with Tailorable Software and Object-orientation, in: Anzai, Y.; Ogawa, K.; Mori, H. (eds): Symbiosis of Human and Artifact - Future Computing and Design for Human-Computer Interaction, Elsevier, Amsterdam 1995, pp. 995 – 1000.
15. Kahler, H.: Developing Groupware with Evolution and Participation - A Case Study, in: Proceedings of the Participatory Design Conference 1996, Cambridge, MA, November 1996, pp. 173-182.
16. Kjær, A. and Madsen, K. H.: Participatory Analysis of Flexibility: Some Experiences, in : Proceedings of the Participatory Design Conference 1994, Chapel Hill, NC, USA, 27-28 October 1994, p. 21-31.
17. Klöckner, K., Mambrey, P., Sohlenkamp, M., Prinz, W., Fuchs, L., Kolvenbach, S., Pankoke-Babatz, U. und Syri, A.: POLITeam - Bridging the Gap between Bonn and Berlin for and with the Users. In Proceedings of

ECSCW'95 (Stockholm, September 1995), Kluwer, Dordrecht, pp. 17-31.

18. McLean, A.; Carter, K.; Lövstrand, L.; Moran, T: User-tailorable Systems: Pressing the Issue with Buttons, in: Proceedings of the Conference on Computer Human Interaction (CHI '90), April 1-5, 1990, Seattle, Washington, ACM-Press, New York 1990, pp. 175-182

19. Malone, Th. W.; Fry, Ch.; Lai, K.-Y.: Experiments with Oval: A Radically Tailorable Tool for Cooperative Work. In: CSCW '92. Sharing Perspectives, Procceedings of the Conference on Computer-Supported Cooperative Work, ACM-Press, New York, 1992, pp. 289-297.

20. Mambrey, P., Mark, G., Pankoke-Babatz, U.: Integrating user advocacy into participatory design: The designer's perspective, in: Proceedings of the Participatory Design Conference 1996, Cambridge, MA, November 1996, pp. 251-260.

21. Mørch, A.: Method and Tools for Tailoring of Object-oriented Applicatios: An Evolving Artifacts Approach, PhD-Thesis, University of Oslo, Department of Computer Science, Research Report 241, Oslo 1997.

22. Nardi, B, and Miller, J.: Twinkling lights and nested loops: Distributed problem solving and spreadsheet development. International Journal of Man-Machine-Studies, Vol. 34, 1991, pp. 69-82.

23. Nielsen, J.: Usability Engineering, AP Professional, New York 1993.

24. Oberquelle, H.: Situationsbedingte und benutzerorientierte Anpaßbarkeit von Groupware, in: Hartmann, A.; Hermann, Th.; Rohde, M.; Wulf, V. (Hrsg.), Menschengerechte Groupware - Software-ergonomische Gestaltung und partizipative Umsetzung, Teubner Stuttgart 1994, pp. 31-50.

25. Olsen, D.R., Foley, J.D., Hudson, S.E., Miller, J., and Myers, B.: Research Directions for User Interface Software Tools, in: Behavior and Information Technology 12. 2. 1993, pp. 80-97.

26. Paetau, M.: Configurative Technology: Adaptation to Social Systems Dynamism. In: Oppermann, R. (Hg.): Adaptive User Support - Ergonomic Design of Manually and Automatically Adaptable Software. Hillsdale, New Jersey 1994: Lawrence Erlbaum Ass. pp. 194 – 234.

27. Pfeifer, A.; Stiemerling O.: *Konfiguration des Informationsdienstes in Groupware*. Proceedings of the 3. Internationale Tagung Wirtschaftsinformatik, 26.-28. Feb. 1997, Berlin, Germany, pp. 263-278.

28. Stiemerling, O.: Anpaßbarkeit von Groupware - ein regelbasierter Ansatz. Diploma thesis (in German), University of Bonn, 1996.

29. Trigg, R. H., Moran, T. P., & Halasz, F. G.: Adaptability and tailorability in NoteCards, in: Proceedings of INTERACT ´87. Stuttgart 1987, Germany, pp. 723-728.

30. Trigg, R.: Participatory Design meets the MOP: Accountability in the design of tailorable computer systems, in Bjerknes, G., Bratteteig, T. and Kautz, K. (eds.), proceedings of the 15th IRIS (Information systems Research seminar in Scandinavia), Larkollen, Norway 1992, pp. 643-656.

31. Wulf, V. and Rohde, M.: Towards an Integrated Organization and Technology Development, in: Olson, G./Schuon, S. (Hrsg.), Symposium on Designing Interactive Systems. Processes, Practices, Methods & Techniques, ACM 1995, pp. 55-65.