in: Proceedings of GROUP '99, ACM-Press, New York, pp. 50 – 60

"Let's see your Search-Tool!" – On the Collaborative use of Tailored Artifacts

Volker Wulf
ProSEC - Department of Computer Science, University of Bonn,
Römerstr. 164, 53117 Bonn,
Germany
volker@cs.uni-bonn.de

ABSTRACT

Groupware applications should be tailorable on different levels of complexity to encourage individual learning and collaborative tailoring activities. A search tool has been developed which offers different levels of tailoring complexity by means of hierarchically organized component languages. Users could create alternative search tools and compound components by themselves. Search tool alternatives and compound components could also be shared among the users. When introducing this tool into an organization of the political administration, it turned out that the users had considerable problems in understanding the functioning of artifacts created by someone else. To ease cooperative tailoring activities, we have implemented features, which allow users to structure, describe, and explore shared components and search tool alternatives. Also we provided means to store and exchange examples for components' use.

Keywords: tailorability, exploration, groupware, component architecture

INTRODUCTION

Tailorability is widely assumed to be a key design requirement for groupware. It allows adapting tailorable aspects of an application during usage to different tasks, personal preferences and group standards. Tailoring can be distinguished from ordinary usage and system development (cf. Henderson and Kyng 1991; Oberquelle 1994; Bentley and Dourish 1995).

Tailoring is carried out by users in their work environment. As the level of tailoring expertise varies among different users, these activities are often carried out cooperatively. Local experts play a crucial role in this cooperation. Being domain experts outside the MIS department, they support other users in adapting their systems to their needs (cf. Mackay 1990, Gantt and Nardi 1992, Nardi 1993, Trigg and Bødker 1994).

As most users of groupware do not have programming skills, the potentials of tailorable application have to be made accessible to this user group. Research on tailorable groupware applications has not yet focused much on end users without programming skills and their cooperation with more skilled users. In the CSCW community much effort has been devoted to the implementation of tailorable architectures and applications (e.g.: Kahler et al. 1999, Bentley and Wasserschaff 1996, Syri 1997, Dourish 1996, Malone et al. 1992). Some of this work provides tailorability only at the level of application programmers. But even those applications, which provide a tailoring interface to end users, have rarely been evaluated in real work environments. Such evaluations are critical, because they show whether users are able to handle the tailoring functionality. Also, the CSCW community has not yet focused on technical support for cooperative tailoring activities. Tailoring is either perceived as an individual activity or it is assumed that cooperation does not need technical support.

By contrary, empirical research on the usage of tailorable single user application tackled some of these issues. According to these findings, applications should be tailorable on different levels of complexity. Such a design turned out being beneficial in dealing with different levels of tailoring expertise among the users, encouraging individual learning and stimulating cooperative tailoring activities (cf. MacLean et al. 1990; Nardi and Miller 1991, Nardi 1993). Moreover, MacLean et al. (1990) implemented rudimentary technical support for collaborative tailoring. Users can send and receive tailored artifacts (buttons representing certain functions) via e-mail.

In this paper, we will present a search tool environment, which supports cooperative tailoring activities by means of tailoring languages of different levels of complexity. Also it supports the sharing of tailored artifacts among users. To evaluate this tailoring environment, it was introduced into an organization of the political administration. It turned out that users had considerable problems to understand the functioning of tailored artifacts created by others. Therefore, the tailoring environment was extended by features, which support users in structuring, describing, and exploring shared components and search tool alternatives. Also we provided means to store and exchange examples for components' use.

Layered Tailoring Environments

Tailorable applications provide tailoring environments. In these environments users create persistent artifacts which determine the tailorable aspects of the application. These artifacts are built by means of tailoring languages.

Tailoring languages can be of different levels of complexity. Henderson and Kyng (1991, pp. 226) distinguish three levels of complexity:

- choosing between alternatives of anticipated behavior,
- constructing new behavior from existing pieces,
- altering the artifact (i.e. reprogramming).

Tailoring languages need to be represented at the user interface.[1] Beyond the languages, tailoring environments may contain additional features, which allow testing, administrating or sharing of tailored artifacts. As users tend to adapt their applications rather infrequently and irregularly (cf. Mackay 1990), these environments should also support the (re-) learning of the tailoring language

Layered Tailoring Languages

In case applications contain tailoring languages of different levels of complexity, one can distinguish two cases. First, tailoring languages of different complexity cover different functions (e.g.: choosing between different printers vs. programming a macro to format documents in a word processor). Second, the different languages allow to tailor the same function on different levels of complexity (e.g.: choosing between different button bars vs. building new button bars). In the following we will focus on the latter.[2] We speak about a layered structure of the tailoring languages in case:

- different languages allow to modify the same function on various levels of complexity,
- languages of higher complexity create tailored artifacts, which modify or extend the less complex tailoring languages.

With layered tailoring languages users design aspects of their tailoring languages themselves. This fact has implications for collaborative tailoring. In case an applications offers only one tailoring language for each tailorable function, users have to learn this language. If local experts support other users, they handle the tailoring language and carry out the tailoring activity fully. In case of layered tailoring languages the situation is different because local experts can use tailoring languages of higher complexity to create tailored artifacts which can be used by other users to carry out less complex tailoring activities. Thus, users without programming skills may be enabled to carry out more sophisticated tailoring activities by dividing the labor in more flexible ways with local experts.

Let see the example of tailoring button bars. In case of a single tailoring language the user would either be able to chose between a given set of button bars defined by the programmers or they were equipped with a tailoring function which allowed to build each time new bars. While the first case does not offer enough flexibility the second case requires that all users master the bar-building function. In case of a layered design of the tailoring environment local experts could build new button bars while end users could just chose between these predefined alternatives.

On an individual level the layered structure stimulates incremental learning of the different languages. (cf. MacLean et al. 1990). To enable cooperative tailoring in the way described before, users without programming skills need to be able to understand the language constructs provided by the local experts (e.g. the button bars built by the local experts). In the following we will briefly present a layered tailoring environment based on a hierarchical component architecture.

A Layered Tailoring Environments based on Components

There are a couple of applications, which contain layered tailoring languages (cf. MacLean et al. 1990, Bentley and Wasserschaff 1996; Fuchs 1998). Nevertheless these approaches leave quite a gulf of complexity between the choice among alternatives and the modification of source code. To bridge this gulf, Stiemerling and Cremers (1998) have implemented a component-based approach to tailorability. Contrary to the traditional application of components in software engineering (e.g. Banavar et al. 1998), they allow for some runtime composition of the components.

Looking at the gulf between alternatives to be chosen from (in this case: of alternative compositions of a function) and modifying source code (in this case Java code), such a component based environment offers quite some additional levels of tailoring complexity. As the environment allows for multiple levels of compound components (hierarchical component architecture), wiring operations to connect components of different levels of abstraction are given (cf. Stiemerling and Cremers 1998). On a lower level of tailoring complexity, the users might just have to connect some compound components, which are meaningful for their tailoring task.[3] On the higher level of tailoring complexity a bigger set of components is available. Some of them may not be meaningful to users because they might just provide alternative technological infrastructures to realize the same function.

The layered structure of component-based languages may encourage cooperative tailoring activities. As the behavior of less complex tailoring functions depends on the activities of other local experts, we have to support end users in understanding how these activities influence the system behavior. Therefore, we will look which additional features have proven to support learning.

Learning of Tailoring Environments

Looking at features which support learning of tailoring languages we can draw on experiences concerning ordinary functions in single user applications. Tailoring

[1] Note that there are different ways to present the same tailoring language.

[2] It offers a wider range of technical flexibility. Moreover, it is the more general case because it is possible to realize the first case by restricting the access of users to certain language layers.

[3] Nardi's (1993) demand for few task-oriented language constructs could be satisfied on this level.

environments for users without programming skills should be designed consistently with the ordinary functionality (cf. MacLean et al. 1990, Oberquelle 1994). Component-based tailoring languages allow to connect a set of components with a set of wiring operations. Thus, the tailoring environment consists out of functions, which select and wire components.

Features which encourage learning of single user applications allow structuring, describing, experimenting with and exemplifying the usage of the functionality (e.g. Carroll and Carrithers 1984, Carroll 1987, Yang 1990, Howes and Payne 1990, Paul 1994). These features are provided by programmers for users. In the case of layered tailoring languages the learning situation is different. End users have to learn additionally about the tailored artifact provided by the local experts. In the following, we will discuss the relevance of the existing features in promoting learning of layered tailoring environments in groupware.

Structuring: Wulf (1999) reports that finding the appropriate function is a major barrier to tailor an application. Survey functions presents all functions according to certain classification schemes. Users get aware of the whole functionality and are supported to find a specific function (cf. Paul 1994). A rather different approach to structure functionality is proposed by Carroll and Carrithers (1984). In training wheels interfaces they distinguish between basic and complex functions. Complex functions are made temporarily inaccessible to avoid frustrating mistakes and to encourage learning of the basic functions. While in both of these cases designers structure the system for users with layered languages, users have to structure tailored artifacts, as well.

Describing: Mackay (1990) found that the lack of documentation of respective functions is a barrier to tailoring. Manuals and help texts are typical means to describe the functionality of applications. A description provided by the vendor informs users about the state transition to which the execution of a function leads. Nevertheless, with layered languages users will have to document their activities, as well. Therefore, Mørch (1997) has suggested that users can modify the design rationales of tailorable functions.

Experimenting: Mackay (1990) and Oppermann and Simm (1994) found that experimentation plays a major role in learning tailoring functions. Nevertheless, Mackay (1990) reports that the fear to break something is a barrier to tailors. Oppermann and Simm (1994) found that the effects resulting from experimenting with tailoring functions are difficult to perceive. “Undo function”, “freezing points”, “experimental data”, and “neutral mode” are features which support users in carrying out experiments with a system’s function. Undo functions allow to reset the execution of (multiple) function while freezing points allow to define certain system states in advance to which users can return to after having tried out other functions. Experimental data are especially created to explore certain functions. A neutral mode replaces the execution of a function by a textual desciption of the effects of this execution. (cf. Yang 1990, Paul 1994). All these features support users in trying out which state transition follows from the execution of a certain function. Still, carrying out experiments can be problematic with multi-user systems like groupware because the state transitions are hard to perceive. Besides, tailored artifacts of higher complexity-levels (e.g. elementary component) might be difficult to test by themselves.

Exemplifying: Examples provided by other users are an important trigger to tailor (Wulf 1999). An animation-machine presents a recorded sequence of interaction (Howes and Payne 1990). Such animation gives an example on how users can apply certain functions. Nonetheless, with layered languages users will have to give examples clarifying the meaning of their self-built artifacts, as well.

POLITeam: The Context of the Study

The POLITeam project is a software development project in which the application partners required technical support for distributed cooperation. The main function of the POLITeam system is to supplement paper work processes with electronic work processes in one German federal ministry and in different bodies of a Northern German state government. To accomplish this, POLITeam offers a shared workspace, electronic circulation folders and E-mail functionality. An already existing groupware system (LinkWorks by DEC) was chosen as a base for development according to specific user and situation requirements (cf. Prinz et al., 1998).

An evolutionary, cooperative approach was used in the design, allowing modifications to be made over time which designers and users reported as beneficial. Within this design process user advocates played a special role. These project members visited the sites regularly, provided support to the users and therefore were able to attain user requirements right away (cf. Mambrey, Mark and Pankoke 1996).

The project started in May 1994 and ended in December 1998, since January, 1995 the system has been installed. While the search tool has been a design issue since the very beginning of the project, the component-based version was developed in the last year of the project.

The study portrayed here was mainly carried out with users of the Representative Body of the state government (SR), located in Bonn. About 30 people work in the Representative Body. They represent the interests of their state especially in the federal legislation process. The body is headed by an undersecretary. The organizational structure of the body mainly consists of sections, which represent state ministries. Most of the sections are one-man-departments with the section manager being the only member. Before the introduction of POLITeam, three typists, who were a central resource, supported these sections. Additionally, there are several administrative sections (cf. Pipek and Wulf 1999).

A Tailorable Search-Tool for Groupware

In the following we will describe the design of component based tailoring environment which implements a search tool for groupware. The search tool basically allows users to specify an inquiry for certain documents stored in the LinkWorks database, start the search engine after the inquiry is specified, display the retrieved documents in different ways and carry out certain operations on the displayed documents (e.g. copy the document).

The design of the search tool and its decomposition into components is based on an empirical investigation on search habits in four different organizations and an evaluation of a first prototype (cf. Kahler 1996). To satisfy different and partly contradictive requirements the search tool is decomposed into six types of elementary components (cf. Figure 1). [4] Four of these component-types are visible at the user interface during normal use, while two of them are invisible during normal use but visible while tailoring (search engine and switches).

Eight different specification components allow building an inquiry mask containing just those search attributes the user typically needs. The start button is a component whose activation finishes the specification of the inquiry and activates the search engine. 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 different result-switches allow to sub-divide the retrieved documents and transfer them to distinct display windows. The implemented switches allow sub-division of the output of the search engine according to the document's location or to the document's name. Two different windows allow to display the retrieved documents either in a normal window presenting the documents' name and further attributes, or in a window which counts the amount of documents retrieved and just presents their number. Three control buttons implement different modes in dealing with documents displayed in the normal window. Either a link or a copy can be created on the users' desktop or the document can be accessed directly.

In order to compose these components the tailoring language contains wiring operations, which allow connecting two different types of ports: input and output 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. Components, which are wired together, are displayed by a connecting line.

Figure 2 shows an example of a search tool in tailoring mode. On the upper left side three different specification components allow the user to define the inquiry (class, name and owner of the document). Lines connect the output ports of the specification components (full blue circles) with the input port of the search engine (empty blue circle). The search button's output port (full red circle) is connected to a second input port of search engine (empty red circle). The search engine is only

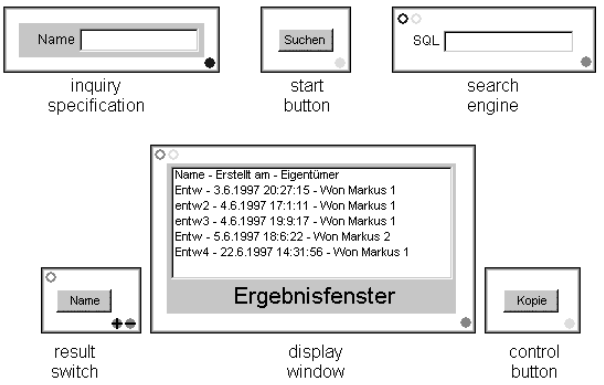


Figure 1: Types of elementary components of the search tool application

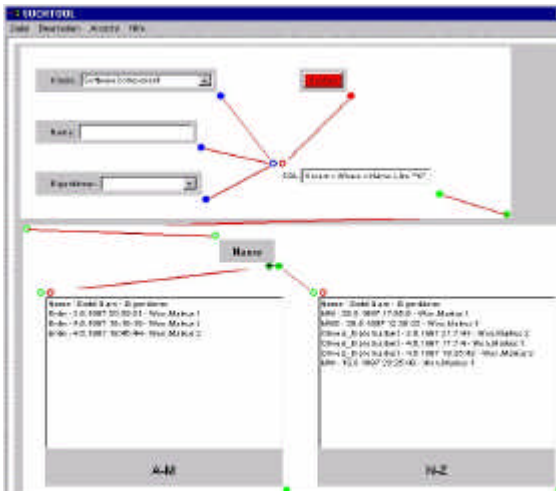
visible in tailoring more. It indicates the inquiry in an SQL-like manner. The output of the search engine is connected (via the compound components input and output ports) to the input port of the result switch (all are green circles). Within the switch the search results are subdivided according to the first letter of the document's name. The documents starting with the letters "a" to "m" are transferred via the left output port of the result switch, while those starting with "n" to "z" are transferred via the right output port of the result switch (both ports are indicated as full green circles). Both output ports are connected to input ports of a display window (empty green circles).

Applying the concept of compound components, tailoring languages of a considerably lower complexity can be created. The search tool presented in figure 2 consists of two compound components (indicated by the white frames). One compound component allows to search (composition of specification components, the search engine and the start button), the other one displays the retrieved documents (composition of a result switch and the two windows). To build the search tool presented in figure 2, users just need to select from the given set of compound components and wire search and display components by means of a single operation (connecting the input and output ports).

On the lowest level of tailoring complexity users can simply choose between alternative search tools. A search tool is a specific compound component, built either from

[4] Won (1998) and Engelskirchen (1999) give a more detailed description of the tailoring language.

elementary components or compound components.



When the search tool is in use, only visible components are displayed at the user interface. Other components, ports, and the wiring statements (connecting lines) are hidden. A pull down menu contains alternative search tools and a menu item, which allows entering into the component based tailoring environment. If the users decide that the given search tool alternatives do not satisfy their requirements, they can simply activate the tailoring environment. In the tailoring mode all components, ports, and the wiring statements (colored lines) of the active search tool are displayed and a second toolbox window appears. This window mainly contains two menus: one pull down menu, which lists all the elementary components and another menu, which lists the compound components. Both lists are represented in alphabetical order according to their names.

To ease the exchange of tailored artifacts among users, we implemented a basic sharing. Whenever a user stores a newly created search tool alternative or a newly created compound component, these artifacts become directly visible in the respective pull-down menus of the other users. Such a design encourages sharing of tailored artifacts in smaller groups.

Evaluation and Extension of the Tailoring Environment

Up till now we have designed layered component languages and a tailoring environment which allows sharing of tailored artifacts. The search tool's tailoring language offers the following layers: elementary components, compound components, and search tools. While all search tools allow choosing among alternatives, the elementary (low-level) components define the most complex tailoring language. The possibility to define compound components is a means to create intermediate language layers. In our case programmers from outside the organization provide the elementary components. By contrary, users and local support staff can tailor the compound components and the search tools. The design of the tailoring environment has to support building and sharing these artifacts.

Evaluating the tailoring environment, the following questions were of especial interest:

- to which extend would users without programming skills be able to tailor the search tool, and
- which division of labor would emerge between the end users and the local experts,
- would end users be able to understand the components, compound components and search tool alternatives provided to them by programmers and local experts,
- how to support the exchange of tailored artifacts between end users and local experts.

Rather than distributing the search tool to a huge population of users from different organizations, we decided to study a small group in depth. To encourage cooperative tailoring, we involved users with different skills and a local expert.

Research Approach

To evaluate the design of the component-based search tool and its tailoring environment, we carried out a workshop and a field test with users from the SR. The workshop was held at the research lab of the University of Bonn. Eleven participants joined that 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 the other users. These users were selected because searching documents was an important part of their work. None of them had programming skills.

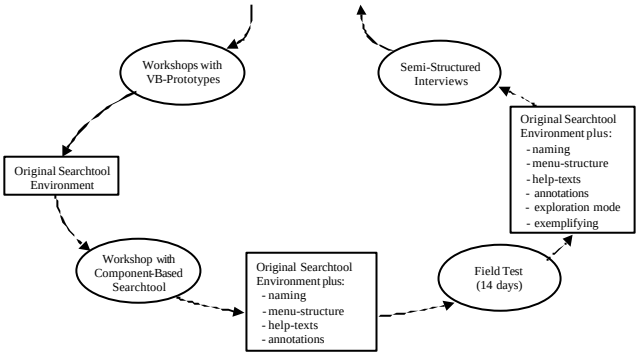


Figure 3: Designing the search tool environment

Also, three user advocates supporting different application partners participated in the workshop. The other participants were members of the POLITeam project involved in the design of the search tool.

After a computer-based presentation of the search tool and its tailoring environment, the four users were asked to apply the tailoring environment to build new search-tools by themselves. Afterwards, issues related to the design of the search tool and the tailoring environment were discussed. The discussion was documented by the project members and transcribed on the day of the workshop.

Based on the results of this workshop the component

language of the search tool and the tailoring environment were extended. In the following we carried out a field study in the SR. Two of the participants of the first workshop had changed their jobs. The remaining ones (the secretary and the clerk who provided system support) and a newly employed clerk from the public relation department of the SR joined the field test. Other users of the SR were asked to provide search permission on their documents eventually needed by the participants of the field test.

We introduced the new search tool environment in a workshop in which three users, one user advocate and three designers participated. The extended tailoring environment was presented and the users were asked to carry out some tasks with it. During the workshop we took written notes and transcribed them directly afterwards.

In the next 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 third version of the tailoring environment was introduced. 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 cooperative 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 in December 1998, we copied all the tailored artifacts for analysis. Figure 3 gives a survey on the design process carried out to improve the usability of the tailoring environment. The following observations are based on an analysis of the different data gained during this process.

Shared Tailored Artifacts and the Division of Labor

In the initial workshop the users already started to discuss how to carry out the tailoring tasks cooperatively. having tried out the tailoring environment, only two of the four users said that they would use it to build their own search tools. The others felt that the graphical interface for connecting components was too complicated to handle without sufficient support. Moreover, these users argued that assembling new search tools would be too time consuming. When searching for documents they feel typically under time pressure which does not allow for tailoring. By contrary the user providing local support was very enthusiastic about the tailoring environment.

During the workshop the users discussed already 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 advocated a tool to distribute newly assembled search tools among the other users. He argued that his job would become much easier with such a tool.

During the field test the implemented sharing mode was therefore well perceived by the users during the field test. 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.

Structuring Components and Tailored Artifacts

To structure the tailoring language(s), the naming and classifying of components and tailored artifacts becomes a central issue. When tailoring during the initial workshop, users had problems to select the elementary components appropriately. These elementary components were labeled by rather design- oriented names. Therefore, the users found it difficult to select the appropriate components from the linear list.

We tackled the problems in three different ways. First, we tried to find more meaningful names for the individual components in cooperation 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.[5]

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 (e.g.: 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.

[5] We simplified the classifcation scheme given in figure 1 by subsuming the search engine and the result switches under one item. The start button was put under the same item as the control buttons.

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 cooperation.

The naming of tailored artifacts became a problem in the field test, as well. For instance, the clerk from the public relations department used the following convention to name search tools she had modified: <old name> (<abbreviation of added components>). This convention was not well understood by the other users. To encourage cooperative 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 collecively tailorable classification schemes seem to be indispensible.

While the classification schemes mentioned so far structure the presentation of the sets of components from the different layers, the users also asked for a context dependent presentation of the subsets of the components. As soon as they touched one of the input ports of a component they wanted to get a context-dependent presentation of the tool-box. This presentation should just list those (compound) components which could be connected to the selected port. In this case the technological nature of the components could be used to generate a context dependent classification scheme automatically.

Describing Components and Tailored Artifacts

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 these documentations.

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.

Contrary to 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 consist of five different text fields: ”name”, “creator,” “origin”, “description”, and “remarks” (cf. figure 4). The “name” field is automatically marked whenever a tailored artifact is created. In the “creator“ field the user who builds a tailored artifact can input 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 functioning 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 cooperative tailoring activities need to develop appropriate conventions.

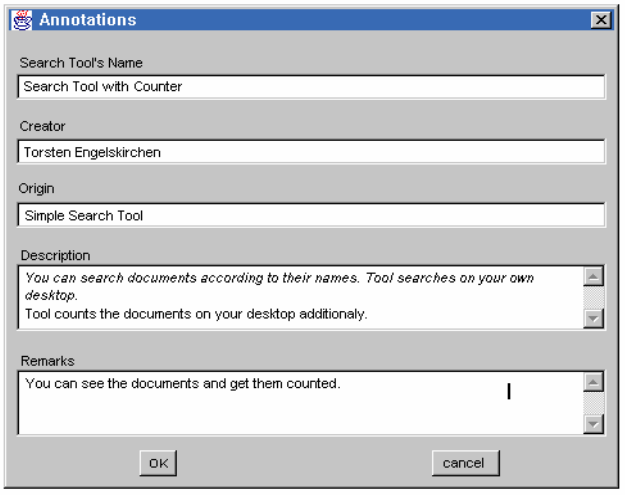


Figure 4: Annotation describing a search tool

Experimenting with Alternative Search Tools

Naming and classification of textual descriptions of tailored artifacts leads to some problems especially in interpersonal usage. Conventions and mutual understanding have to be developed among the different actors. By contrary, experimenting with tailored artifact does not require other users input. In case of single user applications, it just requires to observe the state transitions resulting from a function's execution. In the following, we will investigate how to stimulate experimenting with tailorable groupware.

When trying to find out how an unknown search tool works, users typically switched into the tailoring mode and look at the visualized components and wiring statements. Nevertheless, often they were not able to deduce the functioning of complex search tools because they could not work out the exact meaning of certain components (e.g. output ports of switches) or the outcome of their interplay.

Already during the initial workshop users asked for support in exploring newly assembled artifacts. The user acting as local system support asked to be able to try out newly created search tools:

> „I need to know whether these things do what they are supposed to do."

Nevertheless, the exploration of a tool searching outside the own desktop can lead to disturbances of other users. A statement of the secretary made clear that she would carefully select those users whom to disturb.

> „If I want to know whether the search tool works well on other people's desks, I will send a document to Mrs. P – No! Not to her - but to Mr. S. Then I will search for that document."

Assuming that tailoring is an ongoing activity, even tolerant colleagues will probably not accept permanent interruptions due to other people´s tests.

The problem of exploring assembled search-tools became even more complex in the field test when search permissions had to be explicitly granted. Users who tried out a new search tool, which unexpectedly did not find any documents on other users' desks, had difficulties to judge whether this outcome was due to a wrong understanding of the tool or missing search-permissions.

Therefore, we decide to extend the search-tool environment by an exploration mode. To explore search tools in groupware, other users' documents and desktops have to become visible and accessible. Such an option cannot be realized with other users' real data and desktops, because it would violate their privacy. Consequently other users' desktops, populated by experimental data had to be simulated. Users who want to explore a search tool can take the role of other users and access their simulated desktops.

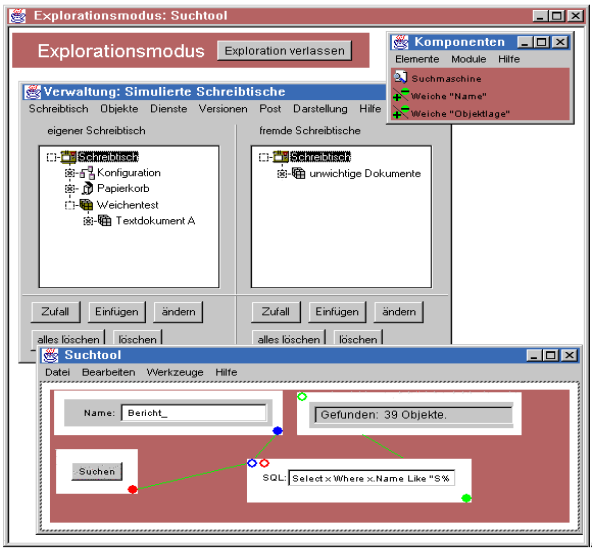
Discussing this concept during the interviews, the user providing local support found the exploration environment useful to test whether a search tool really finds what it was supposed to find. By contrary, the other clerk was quite reserved towards this concept because to her it seemed too complex to 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 neutral mode, freezing points and experimental data.[6] 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 with 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 he started the exploration mode. This search tool was "freezed" during exploration.

Figure 5 shows a screen shot of the exploration mode. The big window in the middle allows populating the simulated desktops with experimental data. To ease the users' understanding of the exploration mode, the menu is designed in a similar way as the main menu of LinkWorks. Only those functions can be activated that

[6] As tailoring in a component-based search environment consists of a long list of various activities, the users regarded a multi-step undo/redo feature as being to complex.



allow creating or modifying data. Moreover, there are further buttons, which allow to populate the simulated desktops more efficiently (e.g. creation of a set of documents by accident). The other two windows (search tool window at the bottom and tool box window in the top right corner) 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 middle window. The button in the top left of the screen allows to leave the exploration mode.

Exemplifying Components' Use

While our tailoring environment supports experiments with given search tools, the question how to support the exploration of compound components or elementary components remains. These artifacts cannot be executed in the search tool environment by themselves. To support users in tailoring we have implemented an option which allows to store a full search tool together with these artifacts. Such a full search tool is supposed to give a characteristic example of how to use the respective elementary or compound component. In this context it can be seen as an executable annotation. As the elementary components were provided by the project team, we have added such an example of a full search tool to teach elementary component. For the compound components we allowed users to "annotate" these tailored artifacts by their own search tool examples.

Activating such an example, the exploration mode comes up. Users can test the search-tool example and deduce the functioning of the tailored artifact from the outcome of these experiments. Besides they can find out about the particularities of the component by replacing it with another one of the same type.

The users found it helpful to see immediately how certain components need to be wired and how to structure the search tool around. One user found a search tool example relevant for his current work and stored it even in the list of alternatives. Due to the fact that this feature was introduced in the end of the field test, we did not really find out whether users would take the extra burden to annotate their compound components by these examples.

Conclusion

Layered tailoring languages are central in promoting individual learning and collective tailoring activities. Contrary to classical tailoring languages, layered languages allow users to design higher-level language constructs by themselves. For two reasons layered tailoring languages are an interesting case for research in CSCW. First, they provide a means to make groupware tailorable to users without programming skills. Second, they are an interesting area to study cooperative tailoring activities.

Hierarchically structured components are a promising approach to implement layered tailoring languages. We applied this technique to develop the component language of a tailorable search tool in gropuware. A field test evaluated the tailorable search tool in real work setting. Our experience indicates that an appropriate language design by itself is not sufficient to secure the usability of the tailoring environment. Based on concepts known to encourage learning in single user applications, we developed features which allow structuring, describing, experimenting with, and exemplifying the usage of components and tailored artifacts. These features had to take the fact into account that with layered tailoring languages users themselves create and exchange tailored artifacts.

Supporting cooperative tailoring activities by sharing the tailored artifact turned out to be a useful concept. However, the sharing mode which was implemented in the case study is only adequate for small groups of users. In larger groups the rising number of compound components and alternative search-tools increases the complexity to choose the appropriate element from the list. In these cases the sharing of artifacts has to become more selective and the establishment of sub-groups of cooperating users has to be supported. Besides sharing, sending of tailored artifacts should be supported in such an environment, as well (cf. MacLean et al. 1990).

Tailored artifacts in layered languages (in this case: compound components and full search tools) have a dual nature. On the one hand they can be seen as shared objects whose meaning has to be articulated between the communities of the producers and the consumers. On the other hand they are part of the formal specification of the application's code. Looking at the means to support articulation between producers and consumers, structuring and describing can be applied to any set of other shared objects. The problems, which we found in the field study, are well known in handling many kinds of shared artifacts. For instance, Mark et al. (1997) and Wulf (1997) report that the naming, classifying and describing was a major problem when working in a shared workspace with documents.

Due to the dual nature of tailored artifacts these problems can be tackled by an experimental execution of the source

code. This provides an additional means to encourage understanding between the producers' and consumers' communities. Yet, experimenting with groupware causes problems, which do not exist, with single user applications. Due to the fact that the effects of a function execution are hard to perceive on the activators workspace new concepts have to be developed. The exploration mode developed in this study gives an example of what is needed.

This research has been carried out on component based tailoring languages for users without programming skills. Nevertheless, the design approach and the features, which support the articulation between producers and consumers, are relevant for component-based applications' development, as well (e.g. Banavar et al. 1998). While the level of complexity of the components and their wiring operations are much higher in that case, there is also the problem that consumers need to understand an artifact created by the producer.

REFERENCES

Banavar, G.; Doddapaneni, S.; Miller, K.; Mukherje, B.: Rapidly Building Synchronous Collaborative Applications By Direct Manipulation, in: Proceedings of CSCW '98, ACM-Press, New York, 1998, pp. 139 - 148

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

Bentley, R.; Wasserschaff, M.: Supporting Cooperation through Customization: The T-Views Approach, in: Proceedings of COOP'96, June 12 – 14, 1996, Juan-les-Pins (Fr.), pp. 241 - 259

Caroll,J. M.: Five Gambits for the advisory Interfacs Dilemma, in: Frese, M.; Ulich, E.; Dzida, W. (eds): Psychological Issues of Human Computer Interaction in the Work Place, Amsterdam 19987, pp. 257 – 274

Carroll, J. M.; Carrithers, C.: Training Wheels in a User Interface, in: Communications of the ACM, Vol. 27, No. 8, 1984, pp. 800 - 806

Dourish, P.: Open Implementation and Flexibility in CSCW Toolkids, PhD Thesis, University College London 1996

Engelskirchen, T.: Exploration anpaßbarer Groupware, M.Sc. Thesis, Department of Computer Science III, University of Bonn 1999 (in preparation)

Fuchs, L.: Situationsorientierte Unterstützung von Gruppenwahrnehmung in CSCW-Systemen, PhD-Thesis, Department of Computer Science, University of Essen, 1997

Grudin, J.: Evaluating Opportunities for Design Capture, in: Moran, J. P.; Caroll, J. M. (eds.): Design Rationale: Concepts, Techniques and Use, LEA, Hillsdale 1996

Gantt, M.; Nardi, B. A.: Gardeners and gurus: Patterns of Cooperation among CAD users, in: Proceedings of CHI '91, May 3-7, 1991, Monterey, CA, pp. 107 - 117

Henderson, A.; Kyng M. (1991): There's No Place Like Home. Continuing Design in Use. in: Design at Work, Lawrence Erlbaum Associates, Publishers, pp. 219-240

Howes, A.; Paynes, S. J.: Supporting exploratory learning, in: Proceedings of INTERACT'90, North-Holland, Amsterdam, S. 881 - 885

Kahler, H.: Developing Groupware with Evolution and Participation: A Case Study, in: Proceedings of the Forth Biennial Conference on Participatory Design (PDC´96), Boston, MA, Nov. 13 - 15, 1996, pp. 173 – 182

Kahler, H.; Mørch, A.; Stiemerling, O.; Wulf, V.: Tailorable Systems and Cooperative Work, Special Issue of Computer Supported Cooperative Work: The Journal of Collaborative Computing, 1999a (in press)

Mackay, Wendy E.: Users and customizable Software: A Co-Adaptive Phenomenon, PhD-Theses, MIT, Boston (MA) 1990

MacLean, 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

Malone, Th. W.; Fry, Ch.; Lai, K.-Y.: Experiments with Oval: A Radically Tailorable Tool for Cooperative Work. in: Procceedings of CSCW '92, ACM-Press, New York, 1992, pp. 289-297

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

Mark, G.; Fuchs, L.; Sohlenkamp, L.: Supporting Groupware Conventions through Contextual awareness, in: : Hughes, J. A.; Prinz, W.; Rodden, T.; Schmidt, K. (Hrsg.), Proceedings of the Fifth European Conference on Computer Supported Cooperative Work - ECSCW '97, Kluwer, Dordrecht 1997, pp. 253-268

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

Nardi, B. A.; Miller, J.: Twinkling lights and nested loops: Distributed problem solving ans spreadsheet development, in: International Journal of Man Machine Studies, vol 34., pp. 161 – 184

Nardi, B. A. 1993: A Small Matter of Programming - Perspectives on end user computing, MIT-Press, Cambridge et al.

Oberquelle, H. (1994): Situationsbedingte und benutzerorientierte Anpaßbarkeit von Groupware. in: Hartmann, A. et al. (eds), Menschengerechte Groupware , Stuttgart, pp. 31-50

Oppermann, R.; Simm, H.: Adaptability: User-Initiated Individualization, In: Oppermann, R. (ed.): Adaptive User Support – Ergonomic Design of Manually and Automatically Adaptable Software. Hillsdale, New Jersey 1994: Lawrence Erlbaum Ass.

Pipek, V.; Wulf, V.: A Groupware's Live, in Proceedings of ECSCW'99, Kluwer, Dordrecht 1999

Prinz, W.; Mark, G.; Pankoke, U.: Designing groupware for Congruency in Use, in: Proceedings of CSCW'98,

ACM-Press, New York, 1998, pp. 373 - 382

Paul, H.: Exploratives Agieren, Peter Lang, Frankfurt/M 1994

Stiemerling, O.; Cremers, A. B.: Tailorable Component Architectures for CSCW-Systems, in: Proceedings of the 6th Euromicro Workshop on Parallel and Distributed Programming, Jan 21-24, 1998, Madrid, Spain, IEEE Press, pp. 302-308

Syri, A.: Tailoring Cooperation through Mediators, in: Hughes, J. A.; Prinz, W.; Rodden, T.; Schmidt, K. (Hrsg.), Proceedings of the Fifth European Conference on Computer Supported Cooperative Work - ECSCW '97, Kluwer, Dordrecht 1997, pp. 157 - 172

Trigg, R. and Bødker, S. 1994: From Implementation to Design: Tailoring and the Emergence of Systematization in CSCW, in: Proceedings of CSCW´94, ACM-Press, New York, pp. 45 – 55

Won, M.: Komponentenbasierte Anpaßbarkeit von Groupware, M.Sc. Thesis, Department of Computer Science III, University of Bonn 1998

Wulf, V.: Storing and Retrieving Documents in a Shared Workspace: Experiences from the Political Administration. In: Howard, S.; Hammond, J.; Lindgaard, G. (eds): *Human Computer Interaction: INTERACT 97*, Chapman & Hall, S. 469-476, 1997

Wulf, V.: Direct Activation: A Concept to Encourage Tailoring Activities, submitted to UIST '99, ACM-Press, New York 1999

Yang, Y.: Current Approaches & new Guidelines for Undo-Support Design, in: Proceedings of INTERACT'90, North-Holland, Amsterdam, S. 543 - 548