J Multimodal User Interfaces (2008) 2: 61–72
DOI 10.1007/s12193-008-0001-5

ORIGINAL PAPER

Web based evaluation of proactive user interfaces

Daniel Schreiber · Melanie Hartmann · Felix Flentge · Max Mühlhäuser · Manuel Görtz · Thomas Ziegert

Received: 15 August 2007 / Accepted: 28 February 2008 / Published online: 23 April 2008

Abstract Usability evaluation of new interface concepts often requires a user study to yield valid results. User studies are however a cost intensive method compared to guideline based usability evaluation. In the AUGUR project we conducted a user study to validate a new interface concept, proactive user interfaces. The method for evaluation we used consists of four steps and relies on leveraging existing tools for realizing each step. These tools can highly automate the data gathering step in the usability study and thereby lower the cost for conducting such a study. In particular they allow for remote participation via a web browser. The obtained data can be analyzed to gain insight into the relation of factors to the three subnotions of usability: efficiency, effectiveness and satisfaction as defined in ISO 9241. We applied the developed method to evaluate the proactive user interfaces we develop in the AUGUR project. In the AUGUR project we aim at augmenting existing user interfaces with proactive and multimodal features to enhance the overall usability of the application. In this paper, we also present the results of the study that shows that proactive augmentations are beneficial for the usability of a user interface and serves as a case study for the application of the evaluation method.

Keywords Remote evaluation methods · Proactive user interfaces · Usability evaluation method · Automated evaluation · Usability testing

D. Schreiber (✉) · M. Hartmann · F. Flentge · M. Mühlhäuser
Telecooperation, Technische Universität Darmstadt,
Darmstadt, Germany
e-mail: schreiber@tk.informatik.tu-darmstadt.de

M. Hartmann
e-mail: melanie@tk.informatik.tu-darmstadt.de

F. Flentge
e-mail: felix@tk.informatik.tu-darmstadt.de

M. Mühlhäuser
e-mail: max@tk.informatik.tu-darmstadt.de

M. Görtz · T. Ziegert
SAP Research, Darmstadt, Germany

M. Görtz
e-mail: manuel.goertz@sap.com

T. Ziegert
e-mail: thomas.ziegert@sap.com

1 Introduction

Usability of software plays an ever more important role, as computers get more involved into daily routines. To successfully integrate the computer interaction with real world tasks often requires new interface concepts. This is a problem, as current usability patterns and guidelines do not apply to new interface concepts for such settings. Therefore the only way to gather reliable usability data is to perform user studies. However, conducting usability studies is a tedious and expensive endeavor. At least a prototype of the interface of the software to test must be implemented and users need to be recruited. Then the users must be observed during the use of the software. Finally, the gathered data must be analyzed to derive some conclusions about the usability.

In the AUGUR project we aim at increasing the usability of existing user interfaces by facilitating the interaction with them. AUGUR provides a platform for the integration and collection of context data and makes use of it in a multimodal and proactive interface, adapting to the individual user's needs at runtime. For evaluating the effects of such a proactive user interface (PUI) on the usability of the user interface we performed a user study. In this paper we describe

the methods we employed for that purpose and the supporting tools for evaluating web interfaces as used in the AUGUR project. The method comprises the setup, execution and analysis of the study. As supporting tools we make use of JavaScript and HTML for creating interface versions and logging during the test. Thus, tests can be taken remotely and no client side installation besides a web browser is required. The resulting data can be analyzed in a standard way to quantitatively compare interfaces with regard to usability. The study we performed for the AUGUR project compares the usability of PUIs with that of a non-proactive baseline interface. The result shows that a proactive extension can increase the usability of an existing user interface. Underlying our method is the definition of usability in ISO 9241 which describes three factors of usability:

- effectiveness,
- efficiency and
- satisfaction.

In Sect. 2 we give an overview of existing approaches for usability studies and explain the theoretical background leading to our approach, introduced in the succeeding section. Section 4 briefly explains the idea of PUIs and shows how a proactive version can be built for any web application by using a JavaScript library. The next section describes how the method for conducting usability studies developed in Sect. 3 has been applied in our own experiment. Section 6 reports the findings of our study and serves as an example of the results which can be obtained by performing studies following the presented evaluation method. Finally, Sect. 7 summarizes and concludes the paper.

2 Related work

Usability evaluation is a well studied topic with many existing methodologies. However, for the study in the AUGUR project we had to meet certain conditions. The evaluation needed to compare an interface using the new interface concepts of PUIs against a conservative, non-proactive baseline, to test whether proactivity had a positive effect on usability. For such a comparison formative approaches meant to be applied during the design process, like e.g. guidelines as in [1], are not suitable.

Other approaches like [2, 3] can be used for summative evaluation. They automatically analyze the interface structure, namely the HTML code of webpages, and generate various usability metrics from it. Examples for such metrics include the download size or the number of pictures on the page. However, as proactivity is a very dynamic feature, it is implausible that these methods would have shown any effect of proactivity, as they only operate on the static features of the HTML code. Additionally, the goal of these approaches is to most closely resemble the judgment of a human expert. As new interface concepts, like proactivity or multimodality are still not widespread, no agreement among experts about how to assess their effect on usability is reached yet. All above methods assess usability only by considering properties of the software itself. They do not consider e.g. the circumstances of use or the user's tasks.

In contrast, the GOMS methodology [4] aims at simulating the user performing a task. The interface is evaluated by analyzing the actions a user needs to perform, to accomplish a task. The result is a rating of the best reachable interface performance. However, there is an important difference between reachable performance and usability as the reachable performance is only achieved by expert users. It is difficult to draw conclusions for the usability for the average user from it. Therefore GOMS cannot be applied for assessing new interface concepts that do not target expert users but the average user, like PUIs.

The most generic approach to compare two interfaces is to conduct a study, where real users perform a realistic task with both user interfaces. This allows to objectively compare interfaces and makes as little assumptions as possible on the nature of those interfaces, and therefore also applies to new concepts like PUIs and can also compare completely different interfaces like a voice and graphical interface. However, user studies are very cost-intensive and can be applied only at a late stage of development, because the application to test must be implemented to some degree to gain realistic results. Still user studies are in our eyes the only way to definitely prove the benefits of a novel interface concept. During the development of PUIs guidelines can be applied, however for the final evaluation of such a new concept a user study is indispensable. The same holds true for multimodal interfaces, where traditional design guidelines often do not apply. Therefore, we decided to evaluate the PUI concept of AUGUR in a user study. To reduce the costs usually involved with such studies we used existing software tools to support the study. In the following section we give an overview of these existing tools.

2.1 Support for automating user studies

The description for methodologies and tool support for usability and interface studies is sparse compared to the detailed methodology computer science brought forward in other areas like software engineering. There is advice on how to set up a study and avoid common pitfalls in evaluating the results, e.g. [5]. However, very few resources exist that are concerned with technical aspects and tool support for such studies. Some initial research into automating user studies was done by [6], suggesting support tools for some steps of a study. However, the focus was on producing records for lab based studies and not for remote studies

like we propose. We consider only those tools that aid in the gathering, processing and analysis of observed usage data. Tools that try to replace real user input, would defy the purpose of a user study.

A very important measure for facilitating user studies is to allow participants to take part in the study remotely, preferably without having to install software. This can be achieved by providing the interfaces to test in form of web pages. This does immediately raise the problem of how to gather fine grained usability data, when the user cannot be observed by an experimentator. The solution is to instrument the website so that the usage data is logged automatically. At least two solutions with this purpose exist, WebVIP [7] and UsaProxy [8]. Whereas the first tool requires access to the source code of the tested application, the second can instrument unchanged applications on the fly. Both report detailed usage data for later analysis at the level of single keystrokes and mouse movements if desired. In AUGUR PUIs are implemented as an extension to existing web applications without access to their code, WebVIP cannot be applied for most PUIs. Creating modified versions for comparison is however not directly supported by UsaProxy and was implemented by us with a JavaScript library. Accompanying WebVIP is the FLUD toolchain which allows to analyze the resulting logs. The tools support the analysis of a website to help a designer improve its usability. They do however not allow to compare two versions of a website.

Both tools produce interaction logs which can be analyzed with regard to usability as defined in ISO 9241. One variable which is easy to measure automatically and does not require explicit feedback is the time needed to perform a certain task. Time corresponds to the efficiency sub-notion of usability in the ISO 9241 specification. Effectiveness, also defined as part of usability in ISO 9241, can be estimated by counting mistakes the user makes during a task, and by counting the total completed tasks. In a study setting where the tasks are fixed, this measure can be obtained automatically from a log review. A completely automated approach for measuring satisfaction the final component of usability from ISO 9241, from log files seems infeasible as it is difficult to observe the user's opinions about a feature without asking for explicit feedback. To overcome this shortcoming of purely interaction log based analysis, we included questionnaires in our methodology. The usage log data can thus be assessed automatically in the three dimensions of usability according to ISO 9241. A comparison of two interfaces can be achieved with statistical methods like ANOVA.

3 Method for user studies

Evaluating new interface technologies and concepts in a user study is a difficult task. To show the effects of new ideas on usability, two comparable interfaces, baseline and enhanced version, must be implemented to such a degree that users can actually use them. Unfortunately, there are no standard datasets for usage data available, which could serve as a baseline and thereby remove the need to gather data for a baseline interface. The first problem is thus to build two interfaces so that usage data can be gathered with them. Once these technical problems are solved, there is the organizational problem of finding enough test subjects to obtain valid results. During the actual test sessions the interactions of the test subjects need to be observed which is also difficult and bears its own problems [9]. Finally, the gathered data has to be analyzed using statistical methods. So the task of conducting a user study can be separated roughly into these four steps.

1. Create baseline and enhanced interface
2. Recruit users
3. Observe interactions of users while using the interfaces
4. Analyze data

We suggest tools and techniques that support each of the above four steps and thereby allow researchers to conduct studies more easily. A common method and tool set is also a step towards comparing results of different researchers which should finally lead to a more thorough understanding of good interfaces.

In principle, there are two basic types of user studies: explorative, where the participants can use the interface freely and non-explorative, task-oriented, where the participants are given a specific task to complete. In the following we concentrate on the latter type. The general setup for a user study compares the two settings baseline and enhanced interface against each other. As the user may be unfamiliar with the task of the study, we suggest to let the user perform the task at least twice albeit with different settings, resulting in within-subject data. We call each such test run a *try* in the following. The data from all users in the same try for different settings can be compared yielding in-between subject data. Thereby every tested condition comprises two varying factors: the number of the *try* and the *setting*.

In the study we conducted, we found a quite large learning effect that may taint within subject data when not treated properly by altering the setting for the different tries through counterbalancing. By doing both, in-between and within subject testing one can also observe the effect of the interface on learnability. To gather meaningful data, we encourage to use realistic application settings of at least some complexity like, e.g. travel booking instead of purely artificial tasks. The contribution of the factors *setting* and *try* to the observed data can be analyzed using One-way ANOVA as explained in Sect. 3.4.

3.1 Creating baseline and enhanced interface

Low fidelity prototypes can be easily created with HTML and JavaScript but can be extended into complete web applications if needed. Another benefit of using HTML for building the prototype is, that every existing web application can be used as a baseline, so that it is easier to compare interfaces for realistic tasks.

Once a HTML version for the baseline interface is available, either newly designed or in form of an existing web application, it can be altered by the use of JavaScript. For existing applications the necessary JavaScript can be inserted via a proxy as shown with UsaProxy in [8], Fig. 1 shows an overview of the setup. The remote server is only accessed via HTTP which allows to use existing interfaces without altering the code. This greatly reduces the effort for creating the enhanced version, as it is sufficient to implement the parts that are relevant for the task of the study. The rest of the application can remain unchanged. Thus, the effects of concepts like proactivity can be evaluated at an early stage with a real user study.

Performing the necessary changes in the baseline interface can be further simplified by using AJAX libraries, e.g. jQuery [10]. With jQuery it is easy to modify the appearance of a web site. It supports interface related features, like changing the layout of HTML elements, hiding or unhiding elements and even animations. The library runs on all major browsers and allows for the simple implementation of vivid interface prototypes. We also used it in our experiment, where we implemented a small library for simulating PUIs on top of jQuery.

Note, that the baseline interface does not necessarily have to reside on a remote server. In our study, we hosted the complete web site on our server, this allows to leave out the proxy server of UsaProxy as explained in Sect. 5. Once the interfaces are ready and setup a few participants should be used to test the whole setup in a pre-study. This is needed to rule out possible problems with the questions or the task.

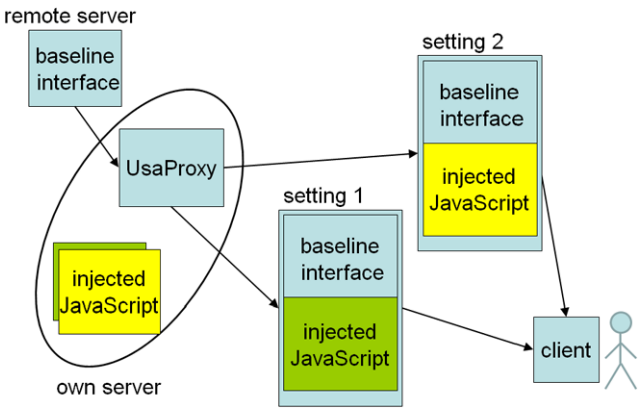

Fig. 1 Setup of the components for observing the user

Usually it will take multiple iterations before all mistakes are corrected, therefore it is advisable to let the test subjects in the pre study take part one by one, thereby maximizing the number of iterations. The participants of the pre-study cannot take part in the real user study, as they are now used to the task and the setup and would therefore taint the gathered data.

3.2 Recruit users

As both the baseline and the enhanced version of the interface are web sites, testing can be done remotely which greatly simplifies the recruiting of test users. A problem with remote testing is that the conditions are not as controllable as during a lab experiment, e.g. there might be unforeseen problems with the web browser of the user. It is also advisable to ask the user explicitly about the circumstances of the test or at least allow for free comment fields where the participants can report anything they found remarkable or disturbing.

3.3 Observing users

As our method allows to remotely participate in the user study, the observation cannot be performed by an on-site experimentator. Thus, data collection has to be done automatically, e.g. via JavaScript or by asking for explicit feedback. UsaProxy [8] is a ready to run solution for the automatic gathering of usage data. This tool allows to collect accurate usage data for any website without client installation. The proxy server injects a JavaScript file into a website and thereby can be used for gathering usage data as well as for introducing the necessary enhancements in the baseline interface as shown in the overview on Fig. 1. The data gathering should be as detailed as possible where data can be gathered automatically, and also comprise technical issues like browser agent and operating system which may help during later analysis. Thus, data like time on task, task completion or errors can be automatically acquired. They are related to the notions of efficiency and effectiveness of usability. To obtain data related to satisfaction one has to rely on questionnaires where the users can give their subjective opinion. For these questionnaires the usual terms apply, that is they should not force the user to one answer and they should not be biased. Besides giving further advise on survey design [11] explains how questionnaires are easily realized using HTML.

3.4 Analyze data

UsaProxy provides accurate timing data which can be analyzed quantitatively. The explicit user feedback from questionnaires leads to nominal data. Further, the errors can be

analyzed by comparing the data the user entered to the correct data needed for completing the task. Special care has to be taken regarding outliers and strange data, as the conditions at the client side cannot be controlled. Also, one is not safe from misbehaving participants which deliberately try to input distorting data or try to test the limits of the interface. However, we think this is not a severe problem for studies where strange data can be easily identified and omitted. The analysis of the data should finally include the observed variables (timing, errors, reported satisfaction etc.) as dependent variables and look for any significant dependencies on the varying factors (*setting* and *try*). Such analysis can be performed with statistical software like SPSS.

4 Proactive user interfaces

In this section, we briefly introduce the main concepts of PUIs (see [12] for a more detailed description) and point out how we implemented the various features for the user study we conducted.

4.1 PUI features

Nowadays applications gain more and more functionality that mostly leads to a decreased usability. We think that this effect cannot be countered by traditional usability engineering but is an inherent design flaw in the current interaction paradigm. Current applications are designed so that the user has full control over the application, which comes at the cost of demanding a lot in terms of cognitive, motoric and visual capabilities. To counter this effect we propose an intelligent interface which disburdens the user by explaining to him how to work with an application, by performing repetitive actions on his behalf, by adapting the interface to his needs, and by suggesting content for input fields. Proactive user interfaces (PUI) aim at combining all these features together with multimodal interaction. Another inherent problem in user interface design is that it is hardly ever possible to build a user interface that fits the needs of all users from novice to expert and provides everybody with the appropriate amount of functionality. PUIs adapt to the user and context maximizing usability for each individual user. We distinguish between four main features of a PUI:

- Interface Adaptation
- Content Prediction
- Task Automation
- Support Mechanisms

For each feature we implemented an action in a JavaScript library. To create a PUI from an existing interface just required to attach these actions to the application. For example, when the user focused a field in the application, an action for suggesting values can be triggered. All actions in the library are defined so that they can work with arbitrary web applications. Thereby it becomes possible to study the effects of PUIs in various domains.

Fig. 2 Example of facilitating the interaction by highlighting the next step

Interface adaptation The interface can be adapted through highlighting the functions the user really needs in the current context, as shown in Fig. 2. A more intrusive approach is to minimize the interface to just these functions. For example, a novice user should only be confronted with a basic set of operations to reduce his cognitive load. However, an expert also needs quick access to more advanced commands depending on his needs. These conflicting demands make it difficult to design a generic interface that fits the needs of all users and thus stresses the need for a user-adaptive interface. There has been a debate for years whether automatic adaptation of the user interface helps or confuses the user [13–15]. Adaptation which is not explicitly asked for by the user is often seen as a flaw that introduces unpredictable behavior and that therefore should be avoided. However, [16] shows that carefully planned adaptation can indeed improve usability and avoid confusion. In any case, the interface adaptation is essential when displaying the application on devices with limited screen space. As PUIs in AUGUR are designed to collect data at a very fine grained level without interfering with normal application usage, the adaptation of the interface can be especially geared towards a single user. This results in a different interface for every user and can even incorporate changing preferences of this user, e.g. caused by learning.

To test the interface adaptation feature we created the *highlight* action in our JavaScript library. This action takes as argument the XPath expression of an element. When invoked this element is emphasized by a bright yellow shade as shown in Fig. 2.

Content prediction and task automation One of the most promising adaptations make use of content prediction where the PUI helps the user performing his task by suggesting how to fill fields. This is again of great importance in mobile settings where the costs of an interaction are much higher than in a desktop setting. For the content prediction we rely on knowledge that is inferred from context information or from previous user interactions. Suggestions get more common with the rise of AJAX technology and can be seen, e.g. at Google Suggest [17]. As extension to these common suggestion technologies, a PUI can offer suggestions for more than one field at once as shown in Fig. 3, thus greatly reducing the interaction costs. A PUI can even go further than

Fig. 3 Suggesting values for several input fields at once

Fig. 4 Example of explanations provided by our PUI

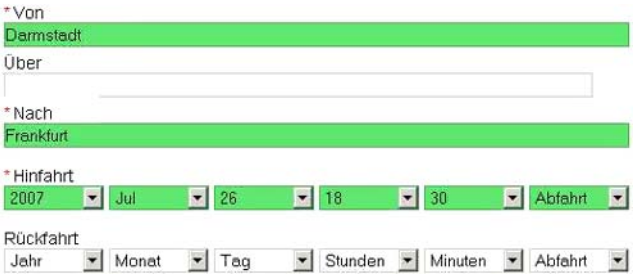

Fig. 5 Application form after the user selected one of the suggestions (see Fig. 3)

only suggesting the values and offer the user to automatically perform actions on his behalf.

For testing the effects of this feature, we implemented a *suggest* and a *prefill* action in our library. The suggest action takes the XPath expressions of the fields for which to suggest along with values as parameter. This allows to display a single suggestion for multiple fields. The visual appearance of the suggestion has been designed after common examples of AJAX suggest functionality (see Fig. 3). The user can thus easily select one of the suggestions, but can also dismiss all suggestions and fill in the form manually. The prefill action takes an XPath expression and value as parameter and automatically fills the designated element with the value. To notify the user which fields were filled by the system after a suggest or prefill action was performed, the elements are highlighted in green (see Fig. 5).

Support mechanism With increasing complexity of software, the number of options the user has to deal with does also increase. Without interface adaptation, the user has to learn to cope with this amount of options and may be forced to find needed functionality by trial and error. This can decrease user-satisfaction, especially if the number of provided options gets too large. The traditional method to cope with this problem is to fall back on classical support mechanisms like manuals or training courses. The most significant drawback of these mechanisms is that they only provide offline help, i.e., the user has to interrupt his work to use them. Although rational behavior would be to read the manual up front as this saves time in total, there are psychological factors preventing users from doing this [18]. Microsoft's Office Assistant presents a more advanced support mechanism [19], as it provides online help according to the user's current task. However, the Office Assistant does not adapt to the user's needs and preferences. Figure 4 shows an example how we incorporated explanations in the PUI, whereby our explanations are as well online as user adaptive.

To reduce the disruptive effect of providing the user with additional support information, it is often of advantage to deliver it over another sensory channel than the task itself [20]. This is the case as the user is forced to switch much or all of his attention to the support information when using the same modality. This can induce structural interferences that lead to negative consequences: errors, decreased accuracy and an increased time for completing the task. Additionally as screen real estate may be limited in a ubiquitous computing setting one has to use supplementary modalities like voice to convey support information.

We implemented a corresponding explanation action in our JavaScript PUI library, but only for gathering qualitative user feedback for this feature. To evaluate explanations properly would require a more complicated task the users were not familiar with. Thus we did not examine the effects this actions had on the times users took for completing the task.

5 Experiment

For our user study of comparing a PUI against a traditional interface, we wanted to make sure that the users were unfamiliar with the specific look of the user interface. Therefore, we chose to create our own mockup of a ticket booking application, instead of using an existing application. We compared a baseline interface against two kinds of PUI interfaces, one simulating the effect of correct proactive support, the other simulating erroneous support of the PUI. Thus, we compared the following three different settings:

- *inactive*: no proactive actions—the baseline interface
- *proactive*: helpful actions were taken by the interface
- *wrong*: erroneous actions were taken by the interface

As task we chose the example of booking a train ticket for a certain day and time with specific properties. Participants were asked to perform the same task twice with a different setting, so we did both, in-between and within subject testing of different settings. The proactive parts were coded with

the help of the JavaScript library for PUIs, so that no client installation was required.

In particular we implemented three PUI actions, described in Sect. 4.

- highlight
- prefill
- suggest

These actions were triggered by user interface events as described below.

To complete the task, the user had to enter data into three web forms. The needed data for successful completion of the task was displayed next to the form as an image (see Fig. 9) to prevent the users from using copy and paste. To obtain within and in-between subject data for all settings the task had to be completed twice with different settings.

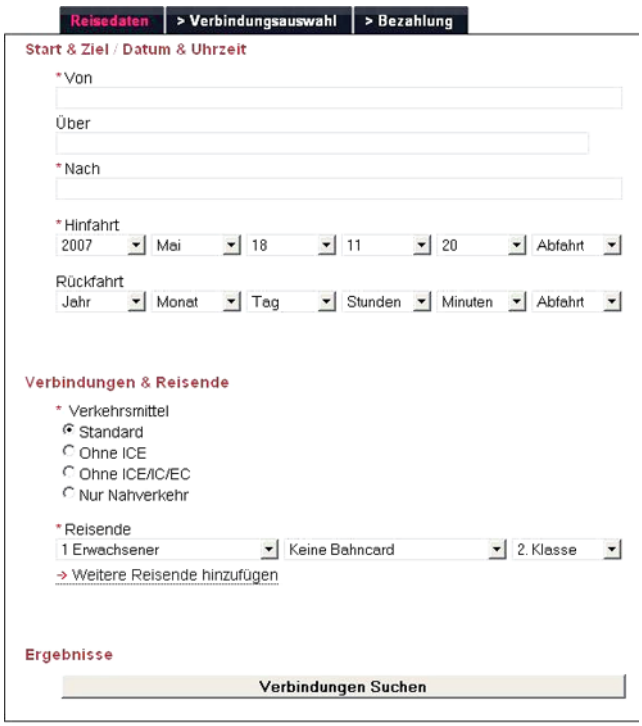

Fig. 6 First form of the task

A short welcome page explains the task and some initial questions about general Internet experience and experience with online ticket booking tasks were asked. Next, the user had to complete the task twice with different settings. To avoid distortion of the results for the second try we used the HTML directive *autocomplete off* in all form fields, avoiding the browsers built in prefill function. After every setting the users also filled out a small questionnaire asking them to rate their subjective opinions of how helpful and how disturbing they perceived the interface on a scale from 1 to 5. They were also asked to rate if they would like to work with such an interface again and how disturbing they think such an interface would be in the long run. The questionnaire after the inactive setting did not contain the questions about the interface's disturbance and helpfulness, because it confused participants in the pre-study. They found these questions difficult to answer if no additional support was given.

The form on the first form required of the user to enter the place of departure and the destination into two text entry fields, to choose the year, month, day and time of travel, and whether to treat this time as arrival or departure time into six drop down boxes (see Fig. 6). In the middle part he could select his means of travel, but no change was required there for completing the target task. Finally, the user had to choose a specific discount option from another drop down menu. In the proactive and wrong setting the interface performed the following actions:

A1 Suggest values for the upper part of the form as shown in Fig. 3. In the wrong setting the suggested values did not completely fit to the task set for the user.
A2 Highlight the drop down for the frequent traveler card shown in Fig. 2 as next action in the proactive setting.
A3 Highlight the button for proceeding to the next step. In the wrong setting a button without any functionality was highlighted.
A4 Help was displayed when the focus entered any of the form fields, see Fig. 4. For this action, no time measures were conducted.

The second form (see Fig. 7), required the user to select the third of the presented train connections. In the proactive and wrong setting the following same action was taken,

Fig. 7 Second form of the task

Fig. 8 Third form of the task

Fig. 9 Reminder for the task in German. English translation: Your task, from: Darmstadt, to: Frankfurt, date: 26.07.2007, departure: 18:30 or later, BahnCard: BC50 2nd class, BC number: 912837465, Ticket: 1 grown-up 2nd class

when the user tried to click on a connection that was marked as "no online booking":

A5 Highlight the text passage "no online booking"

In the third form (see Fig. 8) the user had to enter his frequent traveler card number. Thereby, he was supported by following action in the proactive and wrong setting:

A6 Prefill the frequent traveler card number, whereby the number prefilled in the wrong setting was erroneous,

5.1 Technical setup

The proactive parts of the interface in the proactive and wrong setting were realized by means of the library of PUI actions introduced in Sect. 4. The library was included in all three versions of the web site. The actual modifications were done by small scripts providing the user with correct or erroneous suggestions, highlighting and prefilling. For capturing the usage data we used a JavaScript component. This script registers handlers for all user interface events, allowing to produce a detailed log of clicks and keypresses. The logfile is sent back to the server where it is stored for further processing. This way it is possible to determine accurate timing data of user actions without having to observe them in a laboratory. The data from the questionnaires was processed and stored in a database by a script on the webserver. The whole study was controlled by a single script, that randomly assigned settings to users. Participants only needed to direct their browser to a short URL that we sent them via email. An overview of the setup can be found in Fig. 1. Our script thereby replaced UsaProxy proxy for injecting JavaScript into the delivered web paged.

In total we collected data from 40 participants. Most of them were computer science students or faculty members, but persons without computer science background who would have been hard to reach without a web based test setup were also included.

6 Results

The study compared the usability of a PUI to a non-proactive baseline. Before subjecting the data to statistical analysis, we took care in filtering outliers and strange data. For example, two users used a version of the Safari web browser, where the proactive actions did not work as expected and so we excluded their data for the proactive setting which relies a lot on browser specific features.

The analysis of the data was done using One-way ANOVA with try and setting as factor and the measured times for completing the action and error rates as dependent variables. We verified the homogeneity of variances using the Levene test and used the Welch test instead of normal ANOVA in cases where we could not rely on homogeneity of variances. As common for usability studies we report significance at a level of 0.05.

The format of the obtained data is identical for all studies using the JavaScript logging software. Therefore the analysis can be performed using the same techniques for other scenarios as well. In the following, we describe the obtained results regarding the three notions of usability: efficiency, effectiveness and satisfaction.

6.1 Efficiency

To measure the efficiency of the interface we analyzed the measured total time for all three forms of the study as well as for the subparts of the first form corresponding to the different actions. An overview over the collected timing data is shown in Table 1.

The effects of try and setting on time for the first and third form of the task can be seen in Figs. 10 and 11 respectively. The differences were significant for the total time of the first and third form in both tries. In the second form no significant differences were expected and also not found, as the variants did not differ enough between the settings. There was however a significant difference between the first and second try for each form, which can be explained by the learning that took place. We also analyzed the efficiency with regard to the distinct PUI actions described in Sect. 5. We present here only the analysis of the data for the second try, however results for the first try were comparable.

6.1.1 Suggest

For the suggestion of values in action A1 a significant benefit ($F(2, 51) = 4.441$, $p < .05$) could be observed as shown in Fig. 12. This is the case although the participants were not

Table 1 Recorded timing data. All values in seconds

Setting	Try	Form 1	σ	Form 2 / A5	σ	Form 3 /A6	σ	A1	σ	A3	σ
Inactive	1	64.6	15.5	24.8	12.7	28.6	25.6	47.5	14.0	9.6	6.9
	2	53.5	27.1	9.0	7.4	18.4	19.2	40.6	15.6	26.0	22.8
Proactive	1	56.9	24.8	34.2	28.3	12.4	8.4	29.6	16.4	14.1	13.5
	2	34.6	20.8	7.7	5.2	6.5	4.8	20.2	18.3	4.5	2.8
Wrong	1	91.0	77.1	28.0	25.7	23.0	15.5	56.7	25.0	9.0	6.2
	2	70.0	19.9	51.9	32.7	16.7	6.6	58.6	25.7	7.4	2.3

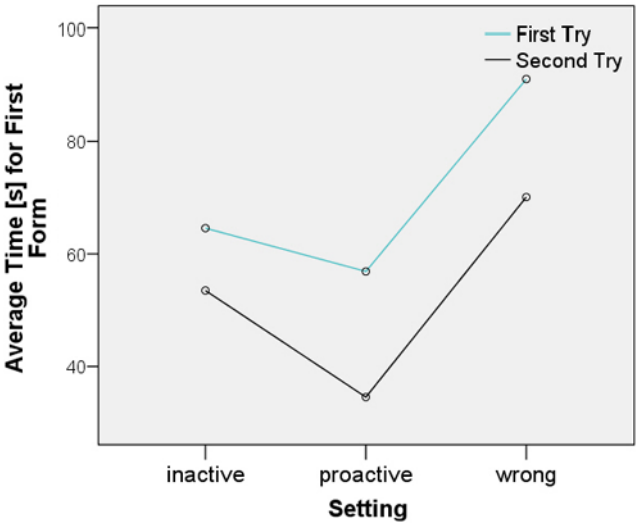

Fig. 10 Effect of the setting and try on the user's performance [s] for the first form

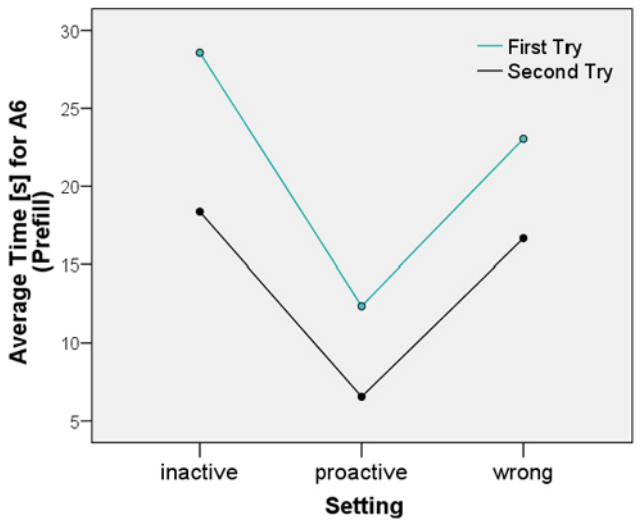

Fig. 11 Effect of the setting and try on the user's performance [s] for form three which corresponds to the action A6

familiar with proactive interfaces. As expected the time for completing this activity with a correct suggestion is smaller

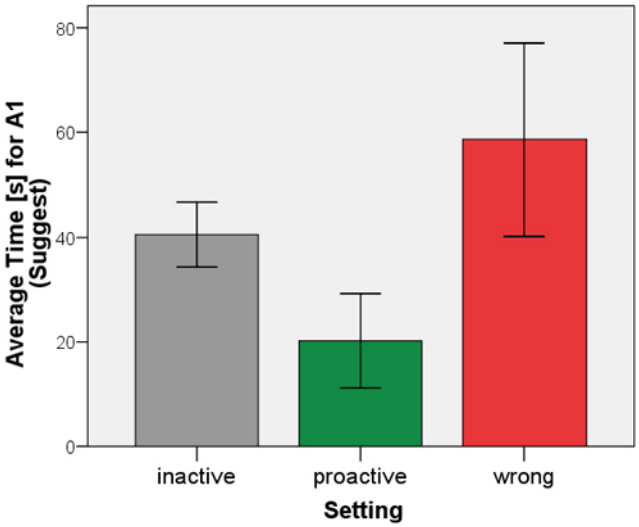

Fig. 12 Average time for the task related to A1

than the time in the inactive setting, while the time in case of a false suggestion is slightly increased.

6.1.2 Highlight

The highlighting action (A5) was not erroneous in the wrong setting, it was the same as in the proactive setting. Thus, we added the results for the wrong setting to the proactive setting. The highlighting of interface elements has no significant effect as shown in Fig. 13. It may have an effect if the highlighting was done more noticeably.

6.1.3 Prefill

In the third form, the number of the frequent traveler card was prefilled with the correct number in the proactive setting and with an erroneous number in the wrong setting. The prefilling with a wrong number does not seem to have a negative effect as Fig. 14 shows. This may be the case as this also causes a highlighting of the field. The structure of this dialog is less obvious than the other dialogues where highlighting had no verifiable effect or the prefilled value serves as an example of how to fill this field.

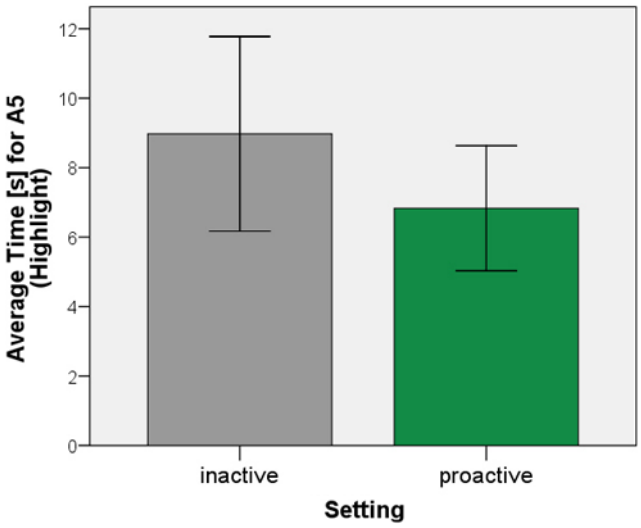

Fig. 13 Average time for completion of A5

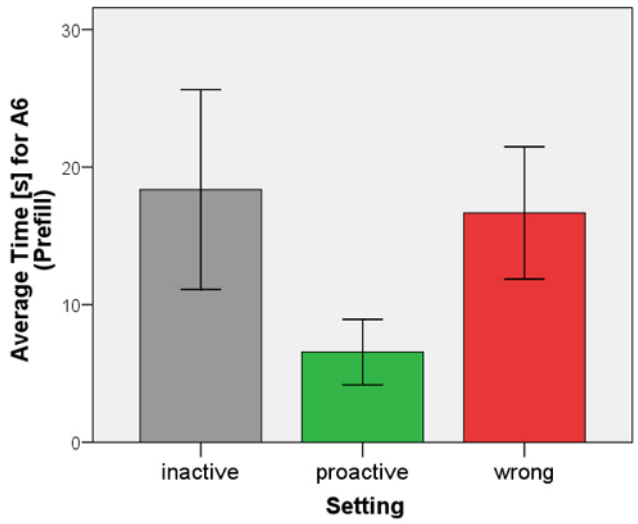

Fig. 14 Average time for completion of A6

6.2 Effectiveness

The effectiveness was measured by counting the number of participants who were able to complete a form without errors. Values for the obtained data for form one and try two can be found in Table 2 that shows the number of participants with the according number of errors for each setting. On average there were only few errors made at each form, so no statistical significance for the influence of the setting or the try on the number of errors was found. However, this may also indicate that there is no significant malus in effectiveness even for a PUI with wrong assumptions. The effect of alternating correct and wrong suggestions may have an effect and needs further study.

Table 2 Recorded error data

Setting	No errors	One error	Two and more errors
Inactive	64%	15%	21%
Proactive	81%	15%	4%
Wrong	42%	17%	42%

6.3 Satisfaction

The participants were asked to rate the support of the user interface ("How much did the system support you in fulfilling your task?") on a scale from 0 to 5, whereby 0 means "not at all" and 5 "very helpful". We found that the proactive interface is perceived as more helpful than the inactive interface when correct support is provided with statistical significance ($t(54) = 4.8$, $p < 0.05$). Our data also indicates that the proactive support is perceived as more helpful even when wrong data is used ($t(40) = 2.34$, $p < 0.05$).

The second question asked was "How disturbing did you find the assistance of the application?" (0 representing "not at all disturbing" and 5 "very disturbing"). This question was only answered by the participants in the proactive setting, as pilot users in the inactive setting were confused by this question as they apparently did not receive any assistance. Participants felt only mildly disturbed for both correct ($n = 23$, $M = 1.261$, $SD = 0.569$) and erroneous ($n = 21$, $M = 2.048$, $SD = 0.925$) support. The next question targeted the users' judgment whether the assistance would disturb in the long run. Like in the previous question the scale ran from "not at all disturbing" to "very disturbing". The data shows that users rated the disruptive effect as rather low, even for the erroneous support (for "correct": $n = 23$, $M = 1.261$, $SD = 0.640$, for "erroneous": $n = 21$, $M = 1.667$, $SD = 0.917$).

The last question, "Would you like to see more applications with this kind of assistance?" was again answered on a scale from 0 "not at all" to 5 "very much", the results for correct ($n = 23$, $M = 3.565$, $SD = 0.901$) and erroneous support ($n = 21$, $M = 2.810$, $SD = 0.763$) show that proactive interfaces are perceived positively by the users. Here again the users rating the interface providing erroneous support was not significantly lower than for the correct support. This shows that users are not irritated or frustrated by wrong suggestions if they are delivered in a proactive fashion and integrated into the interface in a way that does not hamper the normal usage of the application.

There seems to be a trend, that users tended to rate a successful proactive action higher than not taking an action and prefer not taking an action to taking a wrong action as shown in the first column of bars in Fig. 15. Comparing the assistance of the proactive and wrong setting with the inactive setting with regard to satisfaction was not possible with the

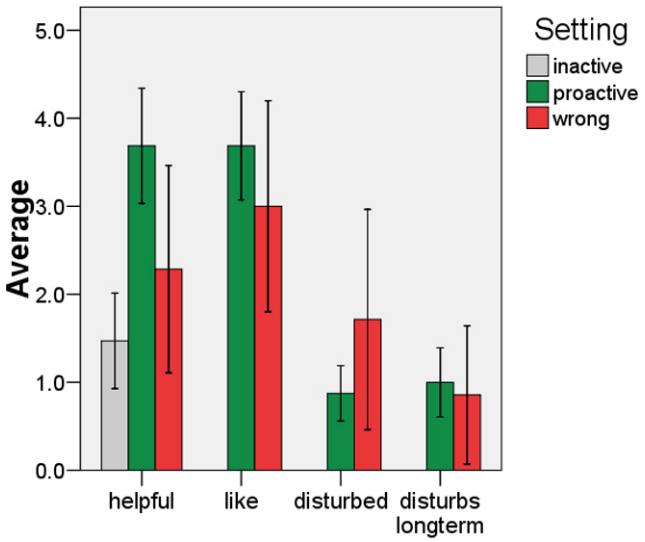

Fig. 15 Subjective ratings

chosen wording. The inactive interface was not considered helpful at all and thereby the users of the pre-study found that the question did not apply.

6.4 Summary

We found significant benefits in efficiency for correct proactive interfaces. And even more important, we did not find dramatic collapses in efficiency nor in effectiveness for erroneous proactive interfaces. This supports our claim, that carefully planned proactive behavior is indeed beneficial, even though it may be sometimes erroneous. For the actions suggest and prefill we found significant improvements in efficiency and they should therefore be used in PUIs. The effect of the highlighting action was not found to be significant. Whether this is due to the concrete implementation, which was maybe not eye-catching enough, is subject to further research.

7 Conclusion

In this paper, we described a process for interface studies and showed how every step can be supported by existing tools. We demonstrated the feasibility of the process by conducting a user study for PUIs following this process. The usage of HTML and JavaScript as a means for creating interfaces has proven effective. It is easy to create new interface concepts and the data collection can be automated using existing tools. The resulting interfaces can be tested remotely, so that one can rely on email as means of recruiting participants. The analysis of the obtained data was done using one-way ANOVA, showing the influence of the setting as well as the try on usability related variables.

Conducting user studies for comparing different versions of websites is not very costly with the appropriate tools and if the analysis is limited to easy quantifiable variables. The outcome of such a study is more suitable to justify researching new interface concepts like proactivity or multimodality than to help improving an existing interface. As the measured variables are not specific to a certain type of interface, they are suitable to assess the effect of new interface concepts and technology. The necessary implementation effort is kept to a minimum by using JavaScript and basing on existing web applications, allowing to perform a user study early in a project.

The used toolset applies also to other scenarios and has been used in the study described in [21]. Also the JavaScript libraries have been tested and integrated even with the complex web applications of SAP. Defining extensions for these applications can also be done with injecting JavaScript through a proxy.

With the described process and analysis method the benefits of PUIs could be shown. Additionally this benefit could by quantized and an estimate of how much time can be saved through PUIs was given. Concerning PUIs the main results were, that suggesting values for multiple fields at once resulted in significant benefit. Prefilling a single field did also increase efficiency of the interface. These two actions are worthwhile considering when implementing PUIs. Further, we showed that erroneous support does not decrease the usability of a PUI.

Acknowledgements We would like to thank SAP Research Darmstadt for supporting our research in the AUGUR project. We also thank all participants of our user study.

References

1. Nielsen J (1999) Designing web usability. New Riders, Indianapolis
2. Ivory MY, Sinha RR, Hearst MA (2001) Empirically validated web page design metrics. In: Proceedings of CHI '01, pp 53–60
3. Palmer JW (2002) Web site usability, design, and performance metrics. Inf Syst Res 13(2):151–167
4. John BE, Kieras DE (1996) Using goms for user interface design and evaluation: which technique? ACM Trans Comput-Hum Interact 3(4):287–319
5. Nielsen J (1994) Usability engineering. Morgan Kaufmann, San Mateo
6. Hicinbothom J, Watanabe M, Weiland WJ, Boardway J, Zachary W (1994) A toolset for systematic observation and evaluation of computer-human interaction. In: CHI '94: conference companion on human factors in computing systems, pp 5–6
7. Scholtz J (2001) Adaptation of traditional usability testing methods for remote testing. In: Proceedings of HICSS '01. IEEE Computer Society, Washington
8. Atterer R, Schmidt A (2007) Tracking the interaction of users with Ajax applications for usability testing. In: Proceedings of CHI '07. ACM, New York, pp 1347–1350

9. Klug T, Mühlhäuser M (2007) Taskobserver: a tool for computer aided observations in complex mobile situations. In: Proceedings of the international conference on mobile technology, applications and systems (mobility), September 2007
10. Resig J, The jQuery Team. jquery. http://jquery.com/
11. Ozok AA (2007) In: Survey design and implementation in human computer interaction, 2nd edn. Lawrence Erlbaum Associates, Hillsdale, pp 1151–1169
12. Hartmann M, Schreiber D, Kaiser M (2007) Task models for proactive web applications. In: Proceedings of WEBIST '07, pp. 150–155, INSTICC Press, March 2007
13. Findlater L, McGrenere J (2004) A comparison of static, adaptive, and adaptable menus. In: Proceedings of CHI '04. ACM, New York, pp 89–96
14. Sears A, Shneiderman B (1994) Split menus: effectively using selection frequency to organize menus. ACM Trans Comput-Hum Interact 1(1):27–51
15. Mitchell J, Shneiderman B (1989) Dynamic versus static menus: an exploratory comparison. SIGCHI Bull 20(4):33–37
16. Gajos KZ, Czerwinski M, Tan DS, Weld DS (2006) Exploring the design space for adaptive graphical user interfaces. In: Proceedings of AVI '06. ACM, New York
17. Google suggest http://www.google.com/webhp?complete=1
18. Carroll JM, Rosson MB (1987) Paradox of the active user. In: Interfacing thought: cognitive aspects of human-computer interaction. MIT Press, Cambridge. Chap 5
19. Horvitz E, Breese JS, Heckerman D, Hovel D, Rommelse K (1998) The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In: Proceedings of UAI '98, pp 256–265
20. Kahneman D (1973) Attention and effort. Prentice Hall, Englewood Cliffs
21. Ries S, Schreiber D (2008) Evaluating user representations for the trustworthiness of interaction partners. In: ReColl workshop at IUI'08. ACM, New York