Empirical Usability Testing in a Component-Based Environment: Improving Test Efficiency with Component-Specific Usability Measures

Willem-Paul Brinkman[1], Reinder Haakma[2], and Don G. Bouwhuis[3]

[1] Brunel University, Uxbridge, Middlesex, UB8 3PH
United Kingdom
Willem.Brinkman@Brunel.ac.uk
[2] Philips Research Laboratories Eindhoven, Prof. Holstlaan 4,
5656 AA Eindhoven, The Netherlands
Reinder.Haakma@Philips.com
[3] Technische Universiteit Eindhoven, P.O. Box 513,
5600 MB Eindhoven, The Netherlands
D.G.Bouwhuis@tue.nl

Abstract. This paper addresses the issue of usability testing in a component-based software engineering environment, specifically measuring the usability of different versions of a component in a more powerful manner than other, more holistic, usability methods. Three component-specific usability measures are presented: an objective performance measure, a perceived ease-of-use measure, and a satisfaction measure. The objective performance measure is derived from the message exchange between components recorded in a log file, whereas the other measures are obtained through a questionnaire. The power of the measures was studied in an experimental setting. Eight different prototypes of a mobile telephone were subjected to usability tests, in which 80 subjects participated. Analyses of the statistical power of these measures show that the component-specific performance measure can be more powerful than overall usability measures, which means fewer users are needed in a test.

1 Introduction

Instead of building an application from scratch, Component-Based Software Engineering (CBSE) focuses on building artefacts from ready-made or self-made components (e.g. pop-up menus, radio buttons, or more complex components such as a spell checker or an email component). Current empirical usability measures do not correspond well with this engineering approach. They do not measure the usability of the individual component, but only its impact on the overall usability (e.g. number of keystrokes, task duration, or questions about the overall ease of use and satisfaction). This indirect way of measuring the usability of a component means that many participants are needed in a usability test. We argue here that component-specific usability measures can be more effective in measuring the usability of an individual component, as they are more focused and therefore require fewer participants in a

usability test. Several authors [9, 19] have suggested that component-specific usability testing might be feasible. They argue that a component can be regarded as an interactive system in its own right with its capacity of receiving input messages, providing users with feedback, and having its own internal state.

In this paper we present a usability testing method that can be used to compare different versions of a component on their usability. The method consists of three component-specific usability measures: an objective performance measure, a perceived ease-of-use measure, and a satisfaction measure. Before describing the testing method, the following section gives an overview of the general characteristics of component architectures on which this method can be applied. After describing the method, an experimental evaluation will be presented, in which the statistical power of the component-specific measures is examined. This section is followed by a discussion of the limitations of the method and its relationship with other empirical usability evaluation methods.

2 Component-Based Interactive Systems

The following three subsections introduce the concepts: control loop, interaction component, and layer. With these concepts it is possible to identify interactive system architectures on which the testing method can be applied, such as the Model-View-Controller (MVC) model [13], PAC (Presentation, Abstraction, Control) model [5] and in particular the CNUCE agent model [17]. The generic architecture described here is based on the ideas of the Layered Protocol Theory [19], which decomposes the user-system interaction into different layers that can be designed and analysed separately.

2.1 Control Loop

Central concepts in the Layered Protocol Theory are the control loop and the accumulation of these control loops. The concept of control loop explains how interaction between users and a system progresses. Interaction is regarded as an exchange of messages between users and the system. Users send messages to the system to change its state. The system sends messages to inform the users about its state. This forms the basis of a negative feedback loop where users compare the received system feedback with their internal mental representation of the state they want the system to be in, the so-called *reference value*. If the reference value and system feedback are not similar, the users may decide to send a message to the system in an attempt to get it in the desired state. When the system receives the users' message, it acts on it, and sends feedback to the users to inform them of the outcome, which again triggers another cycle of the control loop. Once the system is in the desired state, the need for sending messages stops. Therefore, the number of messages sent by the users presents the effort users have made to control the system as each user message indicates a cycle of the loop.

2.2 Architectural Elements

Interaction components define the elementary units of interactive systems, on which behaviour-based evaluation is possible. An *interaction component* is a unit within an application that can be represented as a finite state machine which directly, or indirectly via other components, receives signals from the user. These signals enable the user to change the state of the interaction component. Furthermore, the user must be able to perceive or to infer the state of the interaction component. Therefore, an interaction component should provide feedback. Without the possibility of perceiving the state, the users' behaviour is aimless. Next, it should have a changeable state. A minute label of a radio alarm clock button is not an interaction component on its own because users cannot change it. A behaviour-based measurement of the quality of this label can only be made as part of an interaction component responsible for the minute digits, whose state users can control.

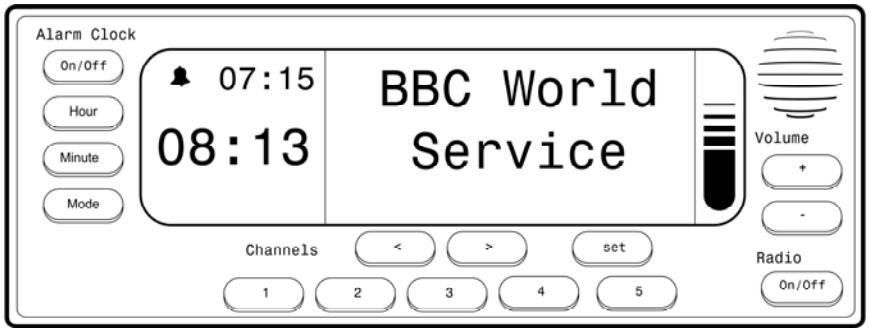

Fig. 1. Front of a radio alarm clock.

The points where input and output of different interaction components are connected demarcate the border between layers. An interaction component operates on a higher-level layer than another interaction component, when the higher-level interaction component receives its user messages from the other interaction component.

Figure 2 illustrates how these concepts can be used to describe a part of the architecture of a radio alarm clock. The three interaction components on the lowest-level layer are responsible for the time (Clock), the selection of the radio stations (Radio Station), and the volume of the sound (Volume). These interaction components receive messages from the users and they send their feedback via the Display component or in case of the Volume component also via the Speaker. Besides sending messages to users as part of their individual control loop, the Clock and Radio Station interaction components also send messages upwards to the higher-level Radio Receiver interaction component. This component fulfils its responsibility in its control loop by sending feedback to the users via the Speak component.

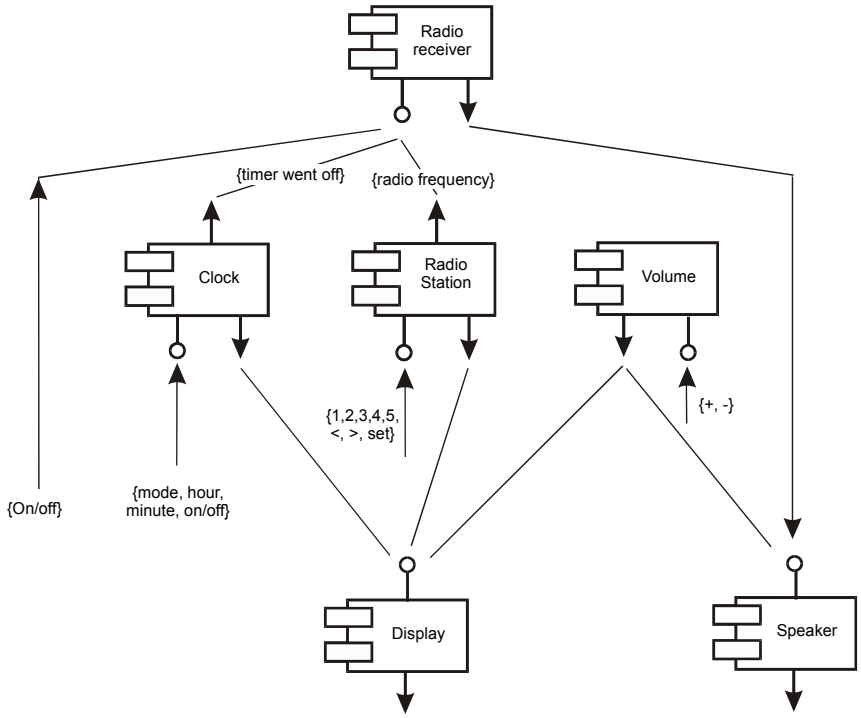

Fig. 2. Compositional structure of a radio alarm clock. The boxes represent components and the arrows the flow of the message exchange between the components.

3 Testing Method

The testing method presented here can be used to test the relative usability difference between two or more versions of a component while the other parts of the system remain the same, e.g. two similar radio alarm clocks that only differ on the implementation of the Radio Station component.

3.1 Test Procedure

The test procedure of the method roughly corresponds to the normal procedure of a usability test. Subjects are observed while they perform the same task with different versions of a system. The task is finished once subjects attain a specific goal that would require them to alter the state of the interaction component under investigation. In advance, subjects should be instructed to act as quickly as possible to accomplish the given goal. As subjects perform the task, messages sent to the interaction

component are recorded in a log file. Once the subjects reach the goal, the recording stops, since new user messages sent afterwards will probably be sent with a new goal in mind.

3.2 Objective Performance Measure

Once the task is completed, the number of user messages received directly, or indirectly via lower-level layers, by the individual versions of the interaction component can be calculated from the log file. This number is put forward as a component-specific performance measure. An earlier explorative study on the affect of foreknowledge [2] indicated that the interaction component version that received the fewest messages is the most usable one. The subjects had to go through the cycle of the control loop less often. Therefore, the number of messages presents the subjects' effort to control the interaction component, provided that the subjects attained only one goal.

The main advantage of the component-specific performance measure is its potential statistical power, meaning that far less subjects are needed in a usability test when data is analysed statistically. The need for a large number of subjects is often one of the reasons why practitioners are unable to run a test because of the time and the cost involved.

Most statistical books that describe statistical testing methods explain in depth the concept of *p*-values but only devote a few paragraphs on power. Whereas the *p*-value in a statistical test is related to the probability of making a type I, or α, error (wrongly rejecting the hypothesis when it is true; for example, predicting a performance difference based on a test while in real life there is no performance difference between two versions of a component) the power of a test is related to a type II, or β, error (failing to reject the hypothesis when it is false). Consider the two distributions in the upper part of Figure 3. The shaded region to the left of the rejection boundary presents the likelihood of making a type β error. The unshaded region on the right of the boundary presents the statistical power of the test, defined as 1- β. In the context of a usability test the power presents the probability of finding a difference between two versions of a component provided there is a difference. A traditional way of increasing the power is by increasing the number of subjects in a test; making the prediction of the distribution more reliable. Another way, however, is to increase the precision of the measuring; making the measure more robust against outside interfering factors, such as possible usability problems the subjects may or may not encounter with other components in the system while completing a task. For parametric statistical tests (e.g. *t*-test, or *F*-test) this means reducing the variance of the sample distribution. Reducing the variance, or in other words making the sample distribution more compact, will also reduce the *p*-value in a statistical test, because the contrast between the two sample groups becomes clearer.

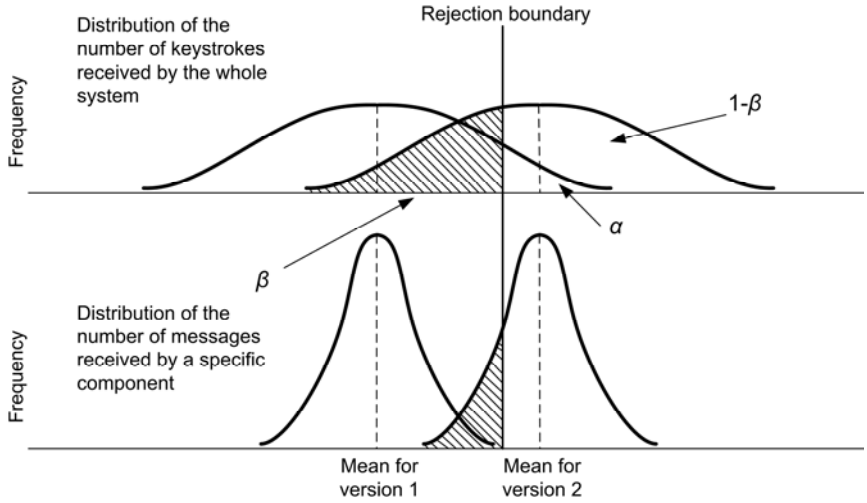

Fig. 3. Performance comparison of two systems implemented with different versions of a component. The variation in the number of keystrokes is larger than the variation in the number of user messages received by the component under investigation, because the first also includes variations caused when users interact with other components, whereas the latter only focuses on the interaction with the relevant component.

The number of user messages a component received directly, or indirectly via lower-level layers, can be a more powerful measure than an overall measure, such as the number of keystrokes, as its variance is smaller. The number of messages received by a component is less likely to be affected by problems located in other parts of the system, whereas overall measures are. In the example with the radio alarm clock, the likelihood that the Radio Station component will receive some extra messages because some subjects have a problem with understanding the Clock component is lower than the likelihood that these subjects make some additional key presses in general. The additional variance, created as subjects try to control other interaction components, is left out in the component-specific measure because of its specific focus. This variance reduction can be apparent in the analysis of lower-level interaction components, but this can apply to higher-level interaction components as well. A low-level message does not always lead to a high-level message. For example, users can still undo a wrong time setting, before the Clock component sends a < *timer went off* > message upwards. Measuring the number of high-level messages will be less affected by variations between subjects interacting with lower-level components. Therefore, the effect of replacing a high-level interaction component with another version can be more obvious in the number of high-level messages than in the number of keystrokes.

The main advantage of making a test more powerful is that fewer samples (subjects) are needed to detect a difference (if there is any) with the same reliability (p-value). Fewer samples are needed because the likelihood of a type β error is

smaller. The lower part of Figure 3 illustrates this point. The shaded region left of the rejection boundary is smaller when samples are more concentrated.

3.3 Subjective Usability Measures

Besides the performance measures, the perceived usability, scaled by subjects, can be used to evaluate the usability of the components. These component-specific questions are expected to be more sensitive than overall usability questions because they help the subjects to remember their control experience with a particular interaction component [4]. The difference between a component-specific and an overall questionnaire is that instead of the system, the name of the interaction component is used in each question. Besides the name of the interaction component, a description, a picture, or even a reference in the system of the interaction component can help to support the subjects' recollection.

Several questionnaires have been proposed in the literature to determine the overall usability of an interactive system. The six ease-of-use questions of the Perceived Usefulness and Ease-of-Use questionnaire [6] seems a suitable small set for a component-specific measure. They make no reference to the system's appearance and are able to capture well-formed beliefs developed by individuals about the ease-of-use after only a brief initial exposure [8]. The component-specific satisfaction questions are taken from the Post-Study System Usability Questionnaire [15], one about how pleasant an interaction component was, and one about how much subjects liked an interaction component. Both the ease-of-use and satisfaction questions use a 7 points answer scale.

4 Experimental Evaluation of the Testing Method

An experiment was conducted to study the method and to test the statistical power of the proposed component-specific measures. The experiment compared prototypes with variations in their usability. The use of predefined usability variations had to emphasise the validity of the usability measures. By seeding *known* usability problems into the prototypes, this experimental set-up ensured that the testing method would identify actual usability problems, and limit uncertainty about whether the measuring had anything to do with usability. In this experiment, all usability variations addressed the complexity of dialogue structures that can be understood in terms of the Cognitive Complexity Theory (CCT) [12]. This theory holds that the cognitive complexity increases when users have to learn more rules.

4.1 Prototypes

A mobile telephone was chosen for the experiment because of its relatively complex system architecture. Furthermore, some of the mobile telephones' interaction components are receptive to well-known and well-documented usability problems. Three interaction components of a mobile telephone were manipulated (printed in

bold type in Figure 4). The three interaction components were responsible for the way subjects could input alphabetic characters (Keypad), activate functions in the telephone (Function Selector), and send text messages (Send Text Message).
For each of these three interaction components two versions were designed. In one version of the Function Selector (FS), the menu was relatively broad but shallow, i.e. all eight options available within one stratum. In the other version, the menu was relatively narrow but deep, i.e. a binary tree of three strata. Users tend to be faster and make fewer errors in finding a target in broad menus than in deep menus [18]. In terms of CCT, the deep menu structure requires the subjects to learn more rules to make the correct choices when going through the deep menu structure. In the more usable version of the Send Text Message (STM) component, the component guided the subjects through the required steps. The less usable version left the sequence of steps up to the subjects. All these steps were options presented as icons, which forced the subjects to learn the icon-option mapping rules. Furthermore, they also had to learn in which order to choose the options.

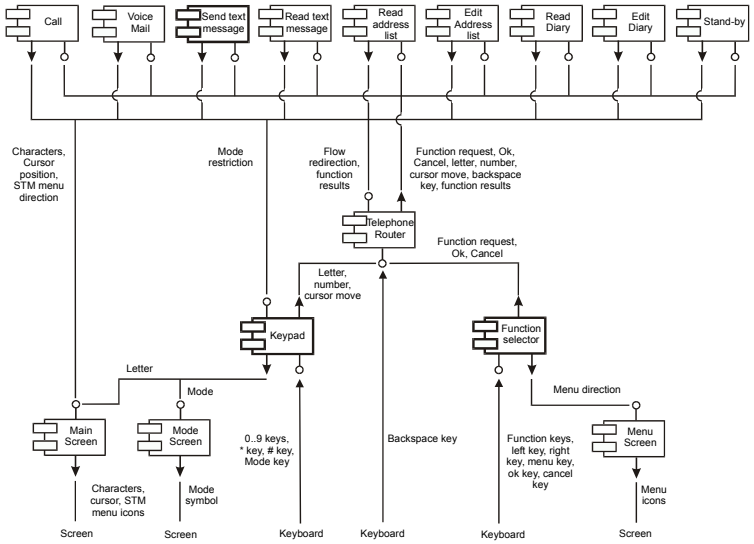

Fig. 4. The architecture of the Mobile telephones (bold interaction components were manipulated in the experiment).

Finally, to enter letters, one keypad version used the Repeated-Key method, and the other version a Modified-Model-Position method. The first is easier to use, because the subjects had to learn one simple rule [7]. It involved having the subjects press the key, containing the letter, the number of times corresponding to its ordinal position on the key (e.g. one time on the "GHI" key for "G"). The other method involved having subjects first press either "*" or "#" key, depending on whether the letter was in the left or right position on the button label and nothing when the letter was in the middle.

This was followed by a press on the key containing the letter (e.g. "*" followed by "GHI" for "G").

Combining these versions led to eight different mobile telephone prototypes. The experimental environment was programmed in Delphi 5, and included PC prototypes of the eight mobile telephones, a recording mechanism to capture the message exchange between the interaction components, and automatic procedure to administer the questionnaire.

4.2 Procedure and Subjects

All 80 participating subjects were students of Technische Universiteit Eindhoven. None of them used a mobile telephone on a daily or weekly basis[2]. The kinds of tasks they had to perform with the mobile telephone were calling to someone's voice-mail system; adding a person's name and number to the phone's address list; and sending a text message. The application automatically assigned the subjects to a prototype in a random order. At the end of the experiment, subjects were asked to evaluate the mobile telephone with the questionnaire on the computer. The computer gave the questions in a random order. After the experiment, the subjects received NLG 22.50 (roughly €10) for their participation.

4.3 Results

The first step in the analysis phase was to conduct multivariate and univariate analyses on the different measures (task duration, number of keystrokes, number of messages received, overall ease-of-use, component-specific ease-of-use, overall satisfaction and component-specific satisfaction). These analyses took as independent variables the different versions of the FS, the Keypad, and the STM component. The results of these analyses can be found in the appendix: Table 3 for the FS component, Table 4 for the Keypad component, and Table 5 for the STM component. The results show in which of the measures a significant effect could be found for the different versions of the component.

Differences in the optimal task performance existed between the versions of the FS and STM component. To compensate for these a priori differences, extra multivariate analyses were performed on the corrected[3] number of keystrokes and messages received measures for the FS and STM component. The results of the analyses can be found in the lower part of Table 3 and Table 5. Unfortunately, no direct way existed to correct the other measures. Still, the corrected keystrokes measure seems an appropriate indicator of how a corrected measure of the task duration would perform;

[2] The experiment was conducted in the autumn of 2000, when a large group of students did not own or use a mobile telephone on a regular basis.

[3] Any additional number of keystrokes or number of messages received created by differences in the optimal task performance between prototypes was subtracted from these samples.

as the time to complete a task was highly correlated (0.91) with the number of keystrokes.

In the second step of the analysis phase, the focus was on the statistical power of the various measures. Because of the relative large sample size (80 subjects), the tests on several measures had a statistical power that approximates to 1. If the experiment were to be repeated, it is almost certain that a significant effect would be found again in these measures. Therefore, a post-hoc power analysis was conducted to calculate the likelihood that a significant difference was found if fewer subjects had participated in the experiment. Various sample sizes were entered in G*Power, a general power analysis program, with the effect size ($\eta^2/(1-\eta^2)$) obtained from Tables 3, 4 and 5.

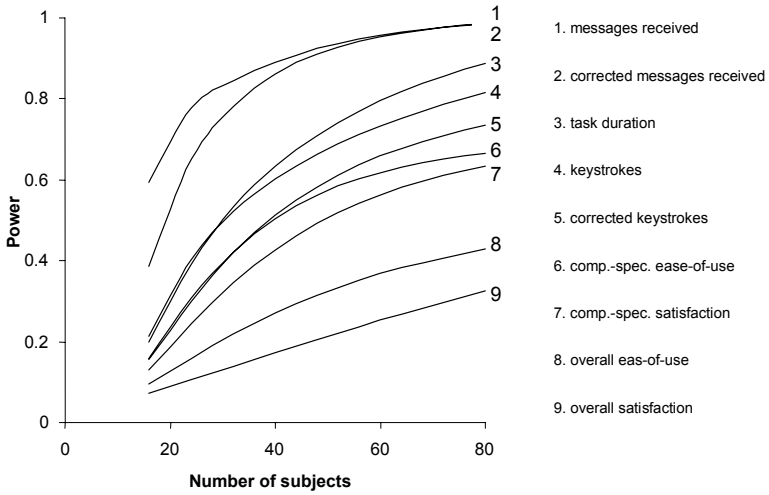

Fig. 5. Average probability that a measure finds a significant ($\alpha = 0.05$) effect for the usability difference between the two versions of FS, STM, or the Keypad components.

Figure 5 presents the statistical power of the different measures averaged over the tests of the three components. The number of messages received was more powerful than the overall objective performance measures such as task duration and the number of keystrokes. For example, if this experiment was set out to find a significant difference with a 60% chance of detecting it, using the number of messages received as a measure would require 16 subjects, whereas the task duration or the number of keystrokes would require 40 subjects —a reduction of 60%.

The effectiveness of the objective component-specific measure is also confirmed by discriminant analyses on the measures. A discriminant analysis does the opposite from what an analysis of variance does. It takes a measure and analyses how well it can predict to which prototype a subject was assigned in the experiment. For each measure per component, a discriminant analysis fitted a linear function that gave the highest number of correct classifications. The function classified the 80 subjects into two groups, one for each version of the component. Although the fitted parameters of

the linear functions are less relevant in this context, the number of correct classifications shows how useful a measure is to discriminate between two versions of a component. In other words, how useful would a measure be in discriminating between low and high usability?

Table 1. Number of correctly classified subjects out of a total of 80 subjects calculated by 18 discriminant analyses. The analyses took the versions of the component as the grouping variable.

Type of Measure	Grouping Variable			Total
	FS	Keypad	STM	
Number keystrokes[a]	55**	52*	42	149
Number of messages received by FS/keypad/STM[a]	71**	52*	63**	186
Ease of use mobile phone	52*	45	50*	147
Ease of use menu/keyboard/STM function	54**	51*	40	145
Satisfaction of mobile phone	45	45	48	138
Satisfaction menu/keyboard/STM function	51*	52*	44	147

Note. Binominal tests, H_0: Number of correct classification = 40.
[a] Corrected for all a-priori differences between versions of the components.
* $p. < .05$. ** $p. < .01$

Table 1 shows the results of the of 18 discriminant analyses. Each analysis was conducted with the versions of the component as a grouping variable. The measure was the independent variable and the versions of the other two components were control variables. The table also shows whether the number of correct classification was significantly higher than the threshold of 40 subjects that on average would be correctly classified by randomly allocating subjects to the groups. Only the linear functions fitted on an objective component-specific measure (corrected number of messages received by the related component) were effective across the three components.

Table 2 shows the results of comparisons on the effectiveness, across the three components, between functions fitted on overall and component-specific measures when it comes to classifying subjects. These comparisons were done on six new variables, two for each type of measure: an overall and component-specific one. A score was assigned to these variables according to the number of times an individual subject was correctly classified. For each subject, the score ranged from zero to three: a zero for no correct classification, a one for one correct classification, a two for two correct classifications, and a three if the versions of all the three components were correctly predicted.

The analyses on the corrected number of keystrokes revealed that 62% (149/240) of the classifications for the versions were correct. This was significantly lower than the 78% correct classifications by functions fitted on the corrected number of messages received. Again, to put the percentage into perspective, note that random allocation would on average link 50% of the subjects with the correct version of the component they had interacted with. Therefore, the relative effectiveness improvement is 32% ((0.78-0.62)/(1-0.5)).

The post-hoc power analysis (Figure 5) indicated that the subjective component-specific ease-of-use and satisfaction measures were on average more powerful than

the subjective overall measures. However, the comparison between the results of the discriminant analyses revealed no significant difference in the number of correct classifications. Looking at Table 1, it seems that the subjective component-specific measures were only ineffective in the case of the higher-level STM component. An unclear reference to this component in the questions might have caused this.

Table 2. Results of Wilcoxon Matched-Pairs Signed-Ranks Tests between the number of correct classification made by discriminant analyses on overall and component-specific measures.

	Correctly classified				
Type of Measure	Overall	Component-Specific	*N*	*T*	*p*
Observed performance	62%	78%	37	3	<0.001
Perceived ease-of-use	61%	60%	62	30	0.907
Perceived satisfaction	58%	61%	61	27	0.308

5 Discussion

To summarize the results, both the power analyses and the discriminant analyses seem to suggest that the objective components-specific measure was more effective than overall measures such as the number of keystrokes. The power analyses also seem to suggest that the subjective component-specific measures were more effective than their overall counterparts. However, the discriminant analyses did not reveal a difference for the subjective measures.

5.1 Limitations

The testing method assumes that the users have to spend the same amount of effort each time they send a message on the level of the interaction component. When high-level interaction components are tested, this assumption is reasonable between the two versions, because the mediating low-level interaction components are the same. However, when the lowest-level interaction components are tested, more attention should be given to this point, as the effort may not be similar. A possible way to solve this problem is by assigning individual weighting factors to the messages [3].

The total number of keystrokes could be more powerful than the component-specific measure when the usability variation of one interaction component influences the number of messages received by another interaction component. This can be caused by three factors: the user, the environment, and the system architecture. For instance, in the mobile telephones equipped with the Modified-Modal-Position method, higher-level String components embedded in the STM and the Edit Address List component (Figure 4) received unintended letters, which the subjects also had to delete. An analysis of variance on the number of backspace messages showed this measure as even more powerful than the number of messages received by the Keypad component [2].

A more practical limitation is the assumption that instrumentation code can be inserted in the software to record the message exchange, which may not always be possible. Fortunately, software tools are being developed to cope with that. For example, the iGuess tool [16] automatically inserts recording code into a Java application without any need for access to the source code.

5.2 Other Empirical Evaluation Methods

Unit Testing. Focussing on the usability of a single component is not new. One of the first usability testing papers presented at the first SIGCHI conference [1] focused on specific components of the Xerox's 8010 "Star" office workstation, such as text selection, icon recognition and the selection of graphic objects. In these kinds of so-called *unit tests*, users are asked only to perform a very limited task that requires interaction with a particular component such as selecting text. For lower-level components this is a powerful testing strategy, since it reduces the variation in the data otherwise caused by the interaction with other components. The drawback is the limited nature of these tasks, as users are not asked to perform the task in the context of a larger, everyday task, such as writing a letter. It assumes that the usability of the lower-level component will not be influenced by other components. However, factors like memory load or inconsistency can create relations between the components that influence the task performance [2]. Instead, applying component-specific usability measures, which presumably are equally powerful, means that users can be asked to perform complex tasks.

Sequential Data Analysis. Often, in sequential data analysis, only lower-level events are recorded, which are first pre-processed into more abstract events before they are analysed. However, these compound messages leave more room for discussion about the system interpretation of the lower-level messages and therefore lack a direct relation with the higher-level components. Extending the low-level messages log file with the system's state makes it possible to construct the system interpretation of lower-level into higher-level messages. Still, it would require the analysis to envision the system response to a low-level message when the system is in a particular state. An example of such an approach can be found in the work of Lecerof and Paternò [14].

Not Event-Based Usability Evaluations. Other usability evaluation methods, such as thinking-aloud, cognitive walkthrough, and heuristic evaluations may in some cases be quicker in assessing the usability of an entire user interface. However, they suffer from a substantial evaluator effect in that multiple evaluators end up with different conclusions when evaluating the same user interface [10]. Usability measures that can be applied automatically leave very little room for such an effect.

Furthermore, current usability evaluation methods have also received criticism for their ineffectiveness in finding real problems that lead to changes in a new version of a system [11]. The introduction of component-specific usability measures may help to overcome this as they lead designers unambiguously to the part that should be changed.

5.3 Exploitation of the Testing Method

In CBSE the creation and the deployment of a component are two independent processes separated over time. In both processes, designers can conduct usability tests and apply the component-specific testing method described in this paper. Identifying and dealing with usability problems in the creation process has the advantage that they do not have to be dealt with each time the component is deployed in a new application. Testing the component in the creation process may require developing a test bed as an actual application might not be available or even unknown when developing a general component library for a specific development environment.

Usability testing once the application is assembled is also needed because only then will it be possible to study the component in the context of the other components. If only *one* version of each component is considered and the aim is to compare the usability of the different components in a single application, the component-specific subjective measures can still be useful. The component-specific performance measure, however, cannot be applied directly since user effort to create messages on different layers may not be the same. A combination of adding weight factors to the messages and correcting for inefficiencies of the user's interaction with higher and lower components has been suggested [3] as a possible solution in that case.

6 Conclusions and Final Remarks

The current study confirms the possibility of testing the usability of individual components, which can be applied in a CBSE environment. The direct benefit of the method seems the statistical power of the component-specific measures. Usability testing of individual components opens the door for sets of usable and re-usable components. Applying these components will increase the chance that the final system will also be usable. However, it will not guarantee this. Components can have an impact on each other's usability [2]. More research is needed to understand how and when outside factors affect the usability of a component, and how system developers should deal with this. Furthermore, the testing method also has the potential for usability testing outside the laboratory. However, the component-specific

performance measure will need to be re-examined because now the evaluator sets the users' goal, which is inappropriate in normal field tests.

References

1. Bewley, W., Roberts, T.L., Schroit, D., Verplank, W.L.: Human factors testing in the design of Xeror's 8010 "Star" Office workstation. Proceedings of CHI'83. ACM Press, New York, NY (1983) 72-77
2. Brinkman, W.-P. Is usability compositional? Doctoral dissertation. Technische Universiteit Eindhoven, The Netherlands (2003)
3. Brinkman, W.-P., Haakma, R., Bouwhuis, D.G.: Usability testing of interaction components: Taking the message exchange as a measure of usability. In Jacob, R.J.K., Limbourg, Q., Vanderdonckt, J. (eds.): Pre-Proceedings of CADUI'2004. Kluwer Academics, Dordrecht, The Netherlands (2004) 159-170
4. Coleman, W.D., Williges, R.C., Wixon, D.R.: Collecting detailed user evaluations of software interfaces. In Swezey, R.W., Post, T.J., Strother, L.B. (eds.): Proceedings of the Human Factors Society - 29th Annual Meeting. Human Factors Society, Santa Monica, CA (1985) 240-244
5. Coutaz, J.: PAC, an object oriented model for dialog design. In Bullinger, H.-J., Shackel, B. (eds.): INTERACT'87. North-Holland, Amsterdam (1987) 431-436
6. Davis, F.D.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Quarterly 13 (1989) 319-340
7. Detweiler, M.C., Schumacher, M.C., Gattuso, N.: Alphabetic input on a telephone keypad. In Proceedings of the Human Factors Society 34th Annual Meeting. Human Factors Society, Santa Monica, CA (1990) 212–216
8. Doll, W.J., Hendrickson, A., Deng, X.: Using Davis's perceived usefulness and ease-of-use instruments for decision making: A confirmatory and multigroup invariance analysis. Decision Sciences 29 (1998) 839–869
9. Haakma, R.: Layered feedback in user-system interaction. Doctoral dissertation. Technische Universiteit Eindhoven, The Netherlands (1998)
10. Hertzum, M., Jacobsen, N.E.: The evaluator effect: A chilling fact about usability evaluation methods. Int. J. of Human-Computer Interaction 13 (2001) 421–443
11. John, B.E., Marks, S.J.: Tracking the effectiveness of usability evaluation methods. Behaviour and Information Technology 16 (1997) 188-202
12. Kieras, D., Polson, P.G.: An approach to the formal analysis of user complexity. Int. J. of Man-Machine Studies 22 (1985) 365-394
13. Krasner, G.E., Pope, S.T.: A cookbook for using the Model-View-Controller user interface paradigm in Smalltask-80. Journal of object-oriented programming 1 (1988) 27-49
14. Lecerof, A., Paternò, F.: Automatic support for usability evaluation. IEEE Transactions on Software Engineering 24 (1998) 863–888
15. Lewis, J.R.: IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. Int. J. of Human-Computer Interaction 7 (1995) 57-78
16. McLeod, I., Evans, H., Gray, P., Mancy, R.: Instrumenting bytecode for the production of usage data. In Jacob, R.J.K., Limbourg, Q., Vanderdonckt, J. (eds.): Pre-Proceedings of CADUI'2004. Kluwer Academics, Dordrecht, The Netherlands (2004) 185-196
17. Paternò, F.: Model-based design and evaluation of interactive applications. Springer, London (2000)
18. Snowberry, K., Parkinson, S.R., Sisson, N.: Computer display menu. Ergonomics 26 (1983) 699-712

19.Taylor, M.M.: Layered protocols for computer-human dialogue. I: Principles. Int. J. of Man-Machine Studies 28 (1988) 175-218

Appendix: Results of Multivariate and Univariate Analyses of Variance

Table 3. Results of two multivariate analyses and related univariate analyses of variance with the version of the Function Selector as independent between-subjects variable.

	Mean		*df*				
Measure	Broad	Deep	Hyp.	Er.	*F*	*p*	η^2
Normal							
Joint measure	—	—	7	66	34.47	<0.001	0.80
Time in seconds	947	1394	1	72	29.56	<0.001	0.29
Number of keystrokes	461	686	1	72	37.72	<0.001	0.34
Number of messages received	67	265	1	72	155.34	<0.001	0.68
Ease of use mobile phone	5.5	4.8	1	72	11.86	0.001	0.14
Ease of use menu	5.6	4.5	1	72	22.33	<0.001	0.24
Satisfaction of mobile phone	4.4	3.8	1	72	4.25	0.043	0.06
Satisfaction of menu	4.6	3.5	1	72	15.96	<0.001	0.18
Corrected[a]							
Joint measure	—	—	2	71	60.96	<0.001	0.63
Number of keystrokes	437	602	1	72	20.27	<0.001	0.22
Number of messages received	52	190	1	72	75.36	<0.001	0.51

[a]Corrected for all a-priori differences between versions of the components.

Table 4. Results of multivariate and related univariate analyses of variance with the version of the Keypad as independent between-subjects variable.

	Mean		*df*				
Measure	RK	MMP	Hyp.	Er.	*F*	*p*	η^2
Normal							
Joint measure	—	—	7	66	4.05	0.001	0.30
Time in seconds	872	1083	1	72	9.44	0.003	0.12
Number of keystrokes	438	537	1	72	10.34	0.002	0.13
Number of messages received	233	271	1	72	13.92	<0.001	0.16
Ease of use mobile phone	5.3	5.0	1	72	1.07	0.305	0.02
Ease of use keyboard	5.6	4.9	1	72	11.13	0.001	0.13
Satisfaction of mobile phone	4.3	3.9	1	72	1.76	0.188	0.02
Satisfaction of keyboard	4.6	3.8	1	72	8.97	0.004	0.11

Note. RK: Repeat-Key, MMP: Modified-Model-Position. Analyses on corrected measures are not presented since these are practically the same.

Table 5. Results of two multivariate analyses and related univariate analyses of variance with the version of the STM component as independent between-subjects variable.

	Mean		*df*				
Measure	Simple	Complex	Hyp.	Er.	*F*	*p*	η^2
Normal							
Joint measure	—	—	7	66	18.16	<0.001	0.66
Time in seconds	523	672	1	72	8.15	0.006	0.10
Number of keystrokes	269	320	1	72	4.56	0.036	0.06
Number of messages received	12	49	1	72	74.18	<0.001	0.51
Ease of use mobile phone	5.0	5.3	1	72	1.15	0.288	0.02
Ease of use STM function	5.1	4.9	1	72	0.35	0.555	0.01
Satisfaction of mobile phone	3.9	4.2	1	72	0.93	0.339	0.01
Satisfaction of STM function	3.9	3.8	1	72	0.26	0.614	0.01
Corrected[a]							
Joint measure	—	—	2	71	20.85	<0.001	0.37
Number of keystrokes	249	289	1	72	2.30	0.134	0.03
Number of messages received	12	34	1	72	26.23	<0.001	0.27

[a]Corrected for all a-priori differences between versions of the components.

Discussion

[Claus Unger] If you have a set of component-specific measures in a system and then decide to change your architecture, how can you move the measures across?

> [Willem-Paul Brinkman] The assumption in this method is that we're only varying one component at a time. However if you want to compare components across very different architectures that's a much more difficult problem. We don't address that with our method.

[Nick Graham] The measure that you're using is the number of messages going back and forth. Wouldn't that tend to say that, for example, the vi editor is more usable than MS Notepad. For example, in vi you can use a regular expression to make many changes, whereas in Notepad you'd have to do each one manually.

> [Willem-Paul Brinkman] The measures are based on participants really performing tasks. A user who does not know vi might generate 1000 messages before they figure out how to make the change. In related areas, we're also looking at assigning different weights to different kinds of messages in the system.

[Bonnie John] A lot of usability errors seem to lie at component boundaries. Your method doesn't seem to address that.

> [Willem-Paul Brinkman] If there is a component that bridges between others, then you can analyse it there. However, if the bridge is made in the user's mind then overall measures rather than component-specific measures will be better. However, if there are mismatches between components, or one of the components is occupying all the user's attention, then the method won't necessarily find these errors.

[Bonnie John] I'm not sure that the questionnaires will allow people to give you valid data. What is your feeling?

> [Willem-Paul Brinkman] The questionnaires are difficult to apply, and in fact we frequently see issues with vocabulary mismatch where the users don't reliably understand which component we're talking about in the questions.