Preliminary version of: Brinkman, W.-P., Haakma, R., & Bouwhuis, D.G.(2008). Component-Specific Usability Testing, *IEEE Transactions on Systems, Man, and Cybernetics - Part A*, vol. 38, no. 5, pp. 1143-1155, September 2008.

Component-Specific Usability Testing

Willem-Paul Brinkman, Reinder Haakma, and Don G. Bouwhuis

Abstract—This paper presents the results of a meta-analysis carried out on the results of six experiments to support the claim that component-specific usability measures are on average statistically more powerful than overall usability measures when comparing different versions of a part of a system. An increase in test effectiveness implies the need for fewer participants in usability tests that study different versions of a component. Three component-specific measures are presented and analysed: an objective efficiency measure, and two subjective measures, one about the ease-of-use and the other about the users' satisfaction. Whereas the subjective measures are obtained with a questionnaire, the objective efficiency measure is based on the number of user messages received by a component. Besides describing the testing method, the paper also discusses the underlying principles such as layered interaction and multiple negative feedback loops. The main contribution of the work described is the presentation of component-based usability testing as an alternative for traditional holistic oriented usability tests. The former is more aligned with the component-based software engineering approach, helping engineers to select the most usable versions of a component.

***Index Terms*—User Centred Design Methodology, Software Testing, Component-Based Software Engineering, Usability Testing.**

I. Introduction

USABILITY testing is an important instrument of the usability engineers' toolkit to analyse an application and make suggestions for usability improvement. Traditional usability tests are holistic in nature, regarding the application as a single entity and producing results about the overall usability such as the number of keystrokes made in a task, the time to complete a task, or more subjective measures obtained via a questionnaire. This holistic approach however is less effective when software engineers follow a Component-Based Software Engineering (CBSE) approach. Instead of building an application from scratch this approach focuses on building applications from off-the-shelf or self-made components (e.g. pop-up menu, radio buttons, or more complex components such as a spell checker or an email component). Rather than speaking of a development project, Horowitz and Lambert [1] speak therefore of an assembly project. They argue that as development time and effort is reduced the focus of these projects turn more towards other activities: such as selection of the right component, integrating these components into a system, and also carrying out value analysis which includes human factor issues. Subsequently, this puts usability testing in a new light, where it can help engineers to develop usable components for future use, select usable components from a component library when developing a new application, or, as part of an integration test, help determining whether for example the user interface provide by the various components do not rely on conflicting mental models.

Focusing on the usability of a single component is not entirely new. For example, one of the first usability testing papers at the first SIGCHI conference [2] focused on specific components of the Xerox's 8010 Start Office workstation, such as text selection, icon recognition and the selection of graphical objects. These so-called unit tests, however, provide less valid results as users are asked to perform a very limited task that only requires interaction with a particular component, e.g. selecting a sentence. In this paper we look therefore at another approach. Instead of scaling down the user task, we examine at a set of component-specific measures that can be used while users interact with the component in the context of a large everyday task, e.g. writing a letter. The component-specific usability measures are part of a testing method that can be used to compare the usability of different versions of a component. Although solely looking at the usability of the individual system parts might not provide the entire usability picture of a system, it gives engineers at least an additional view about the usability of the individual system parts.

A. Motivation

As indicated by several surveys [3]-[5], most usability engineers conduct usability evaluations such as usability tests and regard them as an important method to evaluate an interactive system. Despite this general acceptance as an evaluation method, factors such as time and cost are often mentioned [3],[5] as obstacles for adopting these methods in a project. Field evaluations and usability tests in the lab might be among the most labour intense of these evaluation methods,

Manuscript submitted October 3, 2006.

W. -P. Brinkman is with Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands (phone: +31 (0) 15 2783534; fax: +31 (0) 15 2787141; email: w.p.brinkman@tudelft.nl). He is also with Brunel University, Uxbridge, Middlesex, UB8 3PH, United Kingdom.

R. Haakma is with Philips Research Laboratories Eindhoven, Prof. Holstlaan 4, 5665 AA Eindhoven, The Netherlands (email: reinder.haakma@philips.com).

D. G. Bouwhuis is with the Technische Universiteit Eindhoven, P. O. box 513, 5600 MB Eindhoven, The Netherlands (d.g.bouwhuis@tue.nl).

especially when including quantitative methods. Hypothesis testing, a popular quantitative approach to test whether two means, such as task completion time, are significantly different from one another, can require a large number of participants. Sample sizes such as 20 or 40 participants are not uncommon. These numbers are needed to distinguish with confidence a general trend from variation caused by individual differences. Access to a large group of participants can however be time consuming, expensive or simply not possible. Therefore reducing this need would make any evaluation method more appealing for usability engineers. A good example of this is the popularity of Heuristic Evaluation method [6]. Nielsen and Molich [6] showed that only around five evaluators are need to find around two thirds of the usability problems that would be found by a group of 30 evaluators, a reduction therefore of 83%. Similar asymptotic reduction strategies have been suggested for qualitative oriented usability tests [7]. Quantitative oriented usability tests on the other hand do not aim at finding as many usability problems as possible. Instead they measure the usability and compare it with benchmark values or values obtained from other systems. A strategy of accepting to miss out on a small portion of a long list of usability problems to reduce the sample size is therefore not possible. An alternative however is to increase the strength of the measures, allowing engineers to distinguish with confidence a general trend with fewer participants. This motivated us to study our claim that component-specific usability measures are statistically more powerful than overall measures when comparing different component versions [8], [9]. They reduce the need for large participant groups in quantitative oriented usability tests, which makes these tests more practical and cost effective.

B. Related Work

Usability is defined by the ISO standard 9241-11 as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.” These ultimate usability criteria, effectiveness, efficiency and satisfaction, are often translated in more practical or actual criteria [7] such as system feedback, consistency, error prevention, performance/efficiency, user like/dislike and error recovery [10]. Several methods have been established to evaluate a system on its usability and they can be classified into empirical methods, which include collecting user data, and analytical methods, which do not include collecting user data. Various types of analytical methods currently exist, for example inspection methods such as Heuristic Evaluation [6], SUE Inspection [11], and Cognitive Walkthrough [12], or simulation models such as GOMS [13], or complexity measures [14]. To validate these methods, researchers often rely on results obtained from empirical methods. In this context, usability testing is often regarded as the golden standard [7]. Usability tests are so attractive because of their face validity. To invite users to use a system seems an obvious approach to get an insight into how users use a system. A more extensive discussion about usability testing can be found in [10].

Although recently proposed, component-based usability tests can also be categorised according to two testing paradigms, the Single Version Testing Paradigm (SVTP) and the Multiple Versions Testing Paradigm (MVTP). In the first paradigm, only one version of each component in a system is tested. The focus is on identifying components in the system that hamper the overall usability. SVTP therefore is suitable as part of a software integration test. In the other paradigm, MVTP, multiple versions of only one component are compared while the other components in the system remain the same. This time the focus is on finding the version with the highest usability. MVTP therefore is a paradigm for component development and selection. Different component-based usability testing methods have been proposed for SVTP [15] and MVTP [8]. Both methods use measures derived from recorded user interaction and questionnaires. Whereas in MVTP the recorded interaction data can directly be interpreted [8], in SVTP this data needs to pre-processed taking into account the compositional architecture of the system before comparisons can be made between the components [15]. In this paper we will only focus on the method proposed for MVTP as this can be compared with traditional usability measures, something that is currently not possible for SVTP.

Previous work that has looked at usability in the context of CBSE shows that engineers will have to address a set of additional issues when developing usable systems. For example Hertzum [16] warns that software re-use can cause a series of problems such as a fragmented system image, task gaps, conceptual mismatches, re-keying, scalability problems, and added education and training. The problem of conceptual mismatches has also been demonstrated in the lab [17]. Although components might have been developed in isolation, users are confronted with them simultaneously in a system. Therefore engineers should avoid selecting components with conflicting user interaction protocols. Taylor et al. [18] recognised early on the importance of interaction protocol and have worked on developing a general protocol grammar that describes the way in which possible communication errors are avoided or corrected. Design guidelines to create usable components have also been suggested. For example, Haakma [19] explains that components should provide what he calls both expectation and interpretation feedback to novice users so they can establish appropriate expectations about their interaction, select successful actions, and understand the system interpretation of their actions.

Another human factor issue related to CBSE is mental load. Again, studies in the lab [20] have shown that mental demands made by one component could interfere how users interact with other components. This suggests that engineers should select components which combined mental demand would not overstress the user’s mental capabilities. Usability-supporting architectural patterns has also been proposed [21] to avoid usability problems related to software modularisation for example responding to a user's cancellation command across a series of components. Besides the usability problems associated with CBSE others [22] have also stressed the usability benefits of this approach. They refer to the improvement of system modifiability and maintainability, which increases the system's lifetime, and the ease of keeping it operational.

In this paper we will only focus on component-based usability testing within MVTP. Although other reports have focussed on SVTP [15], or on MVTP within the context of a single experiment [8], or on specific limitations of component-based usability testing [17],[20], here the focus will be on the overall effectiveness of the testing method. By studying component-specific usability measures and overall measures in a series of experiments we will examine their effectiveness and their potential of reducing the number of participants in a usability test. Before describing the testing method, the mathematical principles behind it, and a meta-analysis, the following section gives an overview of the general characteristics of interactive architectures on which this testing method can be applied. The aim of this section is not to propose a new architecture or specification notation for developing usable components. Instead it only attempts is to define a component that can be evaluated. The paper concludes with a discussion on the limitations of the testing method and a comparison with other usability evaluation methods and strategies.

II. Brief Introduction Into Component-based Interaction

Several architectures for interactive systems have been proposed in the literature, such as Model-View-Controller (MVC) model [23], PAC (Presentation, Abstraction, Control) model [24], and the CNUCE agent model [25]. These architectures all have in common the idea of software components that interact with each other by exchanging messages. These messages could be implemented as function or method calls or assigning a value to properties of other components. The message exchange between the components and the user could be implemented as fleshing a light or clicking on a mouse button. Fig. 1 illustrates how, for example, a part of a CD-player (Fig. 2) could be assembled from a Player, a Play List, a Volume, a Speaker, and a Display component. Whereas the lower parts of Fig. 1 interact directly with users by exchanging physical messages, such as pressing a button or presenting a symbol on the display, the Player component also receives user messages indirectly by mediation of other components, such as the <*Play track x*> and the <*Set volume to y*> messages. If a component receives its users' messages by mediation, this component is defined as operating on a higher-level layer than the supporting components. In the case of Fig.1, the Player component would be operating on a higher-level layer than the Play List and the Volume component.

Within this compositional view, a system can be regarded as a set of components, whereby the behaviour of a component such as the Play List component, can be defined as a finite state machine such that C = (S, R, U, v, ω), where S = {'image', 'instant karma!',…, 'give peace a chance'} represents the set of states of the Play List; R = {play, up, down} represents the input alphabet of messages that can be received by this component; U = {play track 1, play track 2,…, play track 20} is the output alphabet of messages that can be sent upwards by this component; v: $S \times R^+ \rightarrow S$ is the state transition function with elements such as (('image', down play), 'instant karma!') for the Play List; and ω: $S \times R^+ \rightarrow U$ is the sent message upward function with elements such as (('image', down play), play track 2). The concept of a component operating on high-level layer than another component means that at least part of the input alphabet are elements of the output alphabet of the other component. In the case of the Play List, U is a subset of R of the Player component.

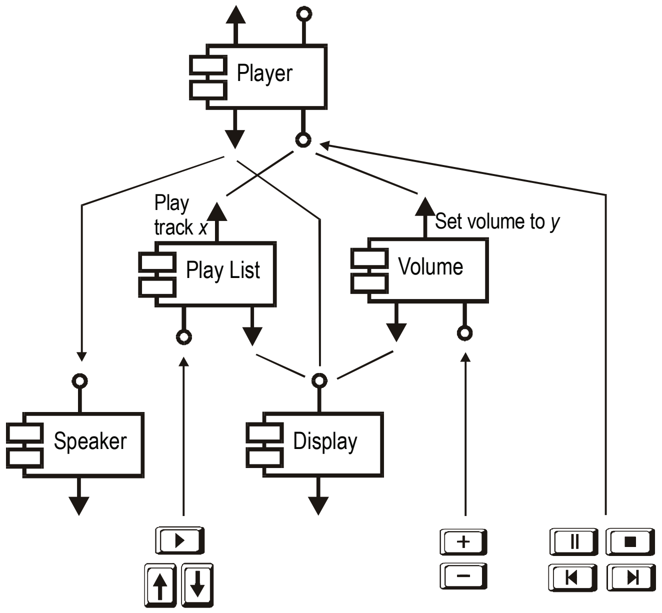

Fig. 1. Part of the compositional structure of a CD-player. The boxes represent components and the arrows the flow of the messages exchange between the components.

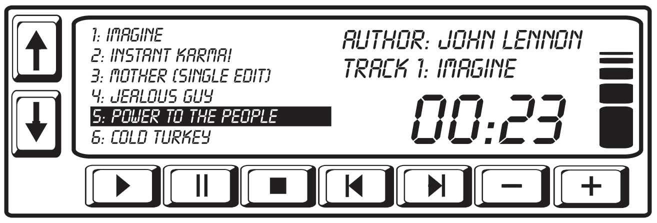

Fig. 2. Front of a CD-player.

The formal specification of a component is sufficient to describe the relevant elements of the component's behaviour that could be tested. However, similar to CNUCE agents [25], the specification could be extended to include the output alphabet of messages sent downwards, and also the sent message downward function. Furthermore, input alphabet could be split into message received from lower and higher-level components, or even consider message exchange between components operating on the same layer.

Besides breaking up a system into layers, the users' mental processes are often also presented as operating in a hierarchy of layers in which higher-level process set goals, or reference values, for lower-level processes (for example, see [26],[27]). Processes that operate on lower-level layers are more physical in nature, such as the coordination of movement of muscle groups. Processes that operate on higher-level layers are more abstract, such as playing the appropriate background music at a party. The Layered Protocol Theory (LPT) [28] brings these compositional views of systems and mental processes together, by suggesting that users interact with a system across several layers by sending messages. In the lowest-level layer, the interaction between mental processes and software components is physical, whereas the messages exchange on higher-level layers is regarded as virtual.

Whereas the layered interaction model explains how the interaction is established, control loops explain the purpose of the interaction. LPT sees the purpose of the users' behaviour as the users' attempt to control their perception, in this case their perception of a system. The users interact with the system because they perceive the system to be in a state other than what they desire it to be in. The control loops are negative feedback loops as users send messages and the system replies with feedback messages, until the users receive feedback that match their intended state, the reference value. Instead of placing the entire system in a single control loop, LPT places every component in a control loop of its own.

Making claims about the usability of a component, based on the execution of a control loop, is only possible if the component has a changeable state that users can perceive or infer. Without feedback and a state, which users can change, there is no control loop and users' behaviour would be aimless. Therefore, to distinguish between testable and non-testable components, the term *interaction component* has been suggested [8] to refer to components that have these properties. In the example of the CD-player, the Player, the Play List and the Volume component are interaction components; while the Speaker and the Display component are not, as they only transform messages. Their usability can only be understood as part of the control loop in which they provide this transformation function, e.g. the control loop of the Player component. Although the definition of an interaction component is rather similar to other definitions of a component such as interactors [25], it does not make reference to the internal organisation of the component. Still engineers might design a system based on another type of definition of what constitute a component, how they are linked and communicate with users. The definition provide only allows engineers to recognise testable component, but leave it open to how engineer slice up a system in a set of components.

III. Testing Method

Having established the concepts of layered interaction together with control loops it is now time to focus on the testing method itself. The testing method can be applied on interaction components that operate in system architectures that allow for control loops, such as MVC, PAC, and especially the CNUCE agent model. The testing method tests the relative usability difference between two or more versions of a component, while the remaining parts of the system stay the same and consequently the type of messages sent to the component. In the case of the CD-player, this could mean testing different versions of the Play List component in the same CD-player. As in ordinary usability tests, the core of the test consists of asking users to accomplish a specific task with different versions of a system. This task, however, should require users to interact with the component that is being tested within the context working (prototype) application. Furthermore, users should also be instructed to complete the task quickly, but also successfully. As a precaution against users ending up trying to solve a task endlessly, a threshold time should be set after which the tester would help them. The threshold time, for example, could be the average task time, obtained in a pilot study, plus three times the standard deviation. This threshold is often used in statistical analyses to find outliers.

As users perform the task, user messages received directly or indirectly by the component are recorded in a log file. The recording stops once the users complete the task successfully, and afterwards the users fill out a questionnaire with component-specific questions about the perceived ease-of-use and the satisfaction of their interaction with the component. The location in the source code for the instructions to record that a message has been received by component depends on the program's architectural and the individual style of the programmer. Potential locations are in the starting lines of a function, or just before or after a function call to the component.

A. Objective Component-Specific Measure

After the test the log file can be studied, and the number of messages the component had received from the user directly, or indirectly via lower-level components, can be counted. Previous studies [8], [29] have shown that the component version that received the fewest messages is the easiest to use, because users had to go though the control loop cycle less often. Therefore, this cycle counter represents the amount of effort users have to invest to get the component to do what they want it to do. The measure only related to the efficiency dimension of the ISO standard 9241-11 usability definition, "The resources expended in relation to the accuracy and completeness with which users achieve goals". Since the participants are given the same goal in a usability test, accuracy and completeness will be the same, making resources expended the only variable to measure. The messages recorded in the log file are the results of an event-chain originated from physical user messages received on the lowest level. Therefore, the effort measured only relates to physical interaction event effort, which is only a small part of the efficiency dimension. Physical interaction event effort, because the measure does not express mental effort, or events that do not result in interaction with the system, for example moving a hand from a mouse to a keyboard. Finally, the measure relates to discrete events, such as keystroke, and consequently is a discrete variable and not a continuous variable such as time to completion a task, or a physiological measure as heart rate variability. Although in the context of the SVTP testing paradigm, the results of an empirical study [15] has shown that the efficiency measure, combined with specific weight factors, was able to provide a valid estimation of the physical interaction event effort users made when interacting with a specific part of a device.

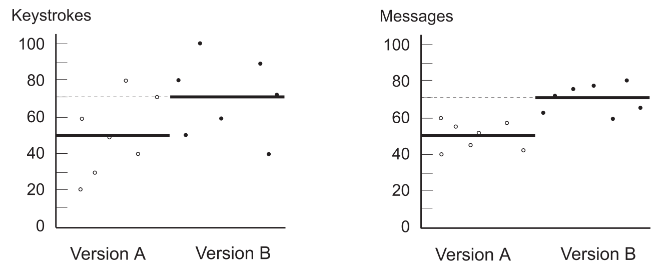

Fig. 3. Plots of the number of keystrokes made (left) and the number of messages received by a component (right) when users performed a task with two versions of a component.

The main advantage of this measure compared to an overall measure such as the number of keystrokes, is its statistical power. In other words, far fewer users are needed when the data is analysed statistically. This would help engineers, as access to a large number of test participants is not always possible, let alone time to conduct such a large test.

The rational for the increased statistical power can be found in mathematical principles underlying hypothesis testing on samples. They show that although using samples might be more practical, it comes with a price, which is uncertainty; uncertainty about how representative the central tendency measures, such as the median or the mean of a sample, are for the whole population. Based on these measures engineers often draw conclusions, and to do this responsibly they need to consider the likelihood of drawing the wrong conclusion, such as concluding that there is a difference when in fact there is no difference. Take for example the hypothetical case that engineers want to know whether on average users can work more efficiently with a new version of a Play List component (version A) than with an existing version (B). They could measure the number of keystrokes two user groups make when completing a task with CD-player A or with CD-player B. The left plot of Fig. 3 shows the hypothetical outcome. Examining the plot, engineers have to decide whether the difference between the group means is caused by the versions of the Play List, or possibly just by random variation, called sampling error. In the latter case, the engineers need to run another test with more users because the data is inconclusive, or accept that there is no obvious, i.e. practical useful, difference which would have been detected in this test. The engineers however, would be in a much more favourable situation to make a decision when the data points within the groups showed less variation and are therefore closer to the group mean, such as in the right plot of Fig. 3. Intuitively it is clear that engineers would conclude much earlier to have found a difference based on the right plot as opposed to the left plot. The power of the statistical test underlying the right plot is said to be larger than that of the left plot. The data in the right plot gives engineers more certainty to claim that it is unlikely that, if the efficiency of two Play List versions were the same, a sample would reveal such a difference. Engineers would therefore reject the so-called zero hypothesis (H_0) of equal means, in favour of accepting an alternative hypothesis (H_1) of unequal means.

The measure, presented in the right plot, is the number of messages received by a component. It gets its power from the reduced variation within the samples compared to the difference between the groups. Textbook statistical analyses, such as the analysis of variance (ANOVA) (see for example [30]), are based on the same principle. For example, the *F*-ratio used in an ANOVA is defined as a ratio (1) of between-sample[1] variation (MS_b) divided by the within-sample variation (MS_w). A large *F*-ratio therefore means a smaller chance (*p*-value) that, if H_0 is true, the samples would show this difference between the groups.

$$F = MS_b / MS_w = (SS_b / df_b) / (SS_w / df_w) \tag{1}$$

Examining how the *F*-ratios are calculated for the number of keystrokes and number of messages in the example gives a clear insight into what is causing this improvement of statistical power. Table I shows the hypothetical number of keystrokes the seven users in the first group made when using version A (X_{i1}) and similar for the second user group that used version B (X_{i2}). With these data points, the sum of squares between groups (SS_b) can be calculated (2), which is based on the deviation between each group mean ($\overline{X}_j$) and the grand mean ($\overline{X}_{..}$).

$$SS_b = n \sum \left(\overline{X}_j - \overline{X}_{..} \right)^2 \tag{2}$$

The group size (n) is 7 and, which makes that SS_b is 7[(50 – 60)^2 + (70 – 60)^2] = 1400. Next is the calculation of the sum of squares within groups (3), which is based on the deviation of

[1] Note that some textbooks refer to this as *treatment* effect instead of *between-group* effect, because it is the difference between the mean of the treatment and the grand mean (2).

each score in a group from its group mean ($\overline{X}_j$), or more formally:

$$SS_w = \sum \left(X_{ij} - \overline{X}_j \right)^2 \quad (3)$$

Therefore SS_w is 2800 + 2800 = 5600. To calculate the *F*-ratio, the degrees of freedom need to be determined. Because this is a comparison between groups with different users, a so-called between-subjects analysis, df_b is equal to the number of groups (*k*) minus 1, and df_w is equal to total number of users (*N*) minus number of groups. The final *F*-ratio is therefore [1400/(2 – 1)] / [5600 / (14 – 2)] = 3. Based on the *F*-ratio and the degrees of freedom it is now possible to calculate the *p*-value, the possibility that a difference has happened by chance. Although the calculation of the actual *p*-value is relatively complex, tables are available to look it up (see for example [30]), or the value can be calculated with software applications such as Microsoft Excel (e.g. using the *F* probability distribution function FDIST(*F*-ratio, df_b, df_w)). In the case of the keystrokes, the *p*-value is 0.10886.

TABLE I
NUMBER OF KEYSTROKES MADE WITH VERSION A AND VERSION B

i	X_{i1}	X_{i2}	$(X_{i1} - \overline{X}_{.1})^2$	$(X_{i2} - \overline{X}_{.2})^2$
1	20	80	900	100
2	60	50	100	400
3	30	100	400	900
4	50	60	0	100
5	80	90	900	400
6	40	40	100	900
7	70	70	400	0
Sum	350	490	2800	2800
$\overline{X}_j$	50	70		

The same procedure can be used to calculate the *p*-value based on the hypothetical number of messages received by the Play List component as presented in Table II. SS_b is the same as before, i.e. 1400. However, SS_w is smaller. It is 314 + 354 = 668. Consequently the *F*-ratio is also larger, namely [1400/(2-1)] / [668/(14-2)] = 25.15. The associate *p*-value for this ratio is 0.0003, giving the engineers more certainty to reject H_0.

TABLE II
NUMBER OF MESSAGES RECEIVED WITH VERSION A AND VERSION B

i	X_{i1}	X_{i2}	$(X_{i1} - \overline{X}_{.1})^2$	$(X_{i2} - \overline{X}_{.2})^2$
1	40	62	100	64
2	60	70	100	0
3	54	75	16	25
4	45	77	25	49
5	51	60	1	100
6	56	80	36	100
7	44	66	36	16
Sum	350	490	314	354
$\overline{X}_j$	50	70		

What becomes clear from this example is that reducing the mean square within groups (MS_w) will improve the chance of detecting a difference between the groups. It is calculated by dividing SS_w by the sum of the degrees of freedom within the groups (df_w). The number of degrees of freedom within each group is a function of the number of samples (*n*). Therefore, the classical and expensive way of improving the power of a test is to increase the sample size in each group as it reduces MS_w, and consequently increases the *F*-ratio. As Fig. 4 illustrates, if there exists a difference, increasing the sample size will increase the likelihood that it will be found (i.e. the statistical power). Another approach, taken by the number-of-messages measure, is to reduce SS_w. This requires that the measure is more robust against outside interfering factors besides the effect established by the difference between the versions of a component.

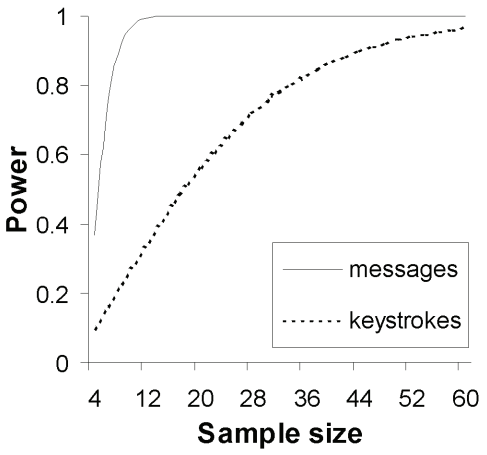

Fig. 4. The likelihood of finding a significant difference between version A and B on an alpha = 0.05 level as a function of the sample size for an analysis on the messages data and on the keystrokes data as presented in Table I and II.

Whereas overall measures, by their very nature, are perceptive to all problems different users may or may not encounter in a system, the number-of-messages measure is mainly perceptive to the problems different users have with a specific component. Or in other words, the variance in the component-specific measure is less likely to increase if some users have a problem with another part of the system. The reduction of SS_{within} can be apparent in the analysis of lower-level as well as high-level interaction. Take for example the Play List component. Variation in the number of Volume '+' and '-' key presses as some but not all users might have problems with setting the volume would also cause variation in the overall number of keystrokes. This however would not effect the component-specific efficiency measure of the Play List component as it ignore these volume events, and only focuses on the play, up, and down buttons events. Whereas for lower-level components component-specific analysis means focusing on a selection of the total lower-level input, for high-level components this means focusing on the messages filtered through by the lower-level components. The filter process reduces the variance in the number of high-level messages. Users' problems with a lower-level component are likely to be confined to that component and do not have to result in high-level messages. For example, some users would need more up and down scrolling through the play list than others, however pressing the play button would still result in a single high-level <*play track x*> message to the Player component.

B. Subjective Component-Specific Measures

The questionnaire at the end of a usability test provides the data for the two subjective usability measures about the component-specific ease-of-use and satisfaction. Whereas overall questions allow users to express their feeling of progress towards the overall task goal that is achieved with the entire system, component-specific questions allow users to express their feeling of progress towards the sub-goal that is achieved with a component. These component-specific measures are expected to be statistically more powerful than overall usability questions because they help the users to remember the control experience with the particular interaction component [31]. So far, however, attempts to show this empirically have only been partly successful [8]. On the other hand, an examination of several empirical studies concluded that the component-specific ease-of-use measure can obtain an acceptable level of reliability and validity, although not all cases, and continued to suggest that with component-specific measures at least part of the usability of a product can be studies on a detailed, compositional level [32].

Several questionnaires have been presented to determine the overall usability of a system in the literature. The six ease-of-use questions of the Perceived Usefulness and Ease-of-Use questionnaire [33] have been shown to be a suitable small set of questions [8]. They make no reference to the system's appearance and are able to capture well-formed beliefs after only a brief initial exposure [34]. The set of questions consist of statements such as "My interaction with [name] would be clear and understandable" [33]. Where traditionally the name of the system would replace the "[name]" section, component-specific questions are created by replacing this section with the component's name. Besides the component's name, a description, a picture, or even a reference in the system of the component can help to support the recollection of the users when they complete the questionnaire.

The component-specific satisfaction questions are taken from the Post-Study System Usability Questionnaire [35], one about how pleasant a component was, and one about how much the user liked using the component. Both the ease-of-use and satisfaction questions use a 7 points answer scale.

IV. Examining the Effectiveness

Previous reports [8], [9] have only studied the effectiveness of component-specific measures in the context of a single experiment. However a single experiment rarely provides the final answer on any empirical question. The approach therefore taken in this paper is an empirical meta-analysis. This analysis examines the effectiveness of component-specific versus overall usability measures in 12 component tests that were carried out as part of six usability experiments that looked into the compositionality of usability [29]. Meta-analysis are often conducted in disciplines such as psychology, and some have also been carried out on software engineering data, for example Shaw [36] presented a meta-analysis of 25 studies on Group Support Systems, and recently Haque and Srinivasan [37] applied a meta-analysis on 16 studies and successfully showed the learning effect of virtual reality surgical simulators. Meta-analysis has also been conducted to study software analysis methods. Miller [38] for example conducted a meta-analysis on five experiments that studied code reading versus functional software testing, whereas Hayes [39] conducted a meta-analysis on five published studies of software engineering inspection methods. Attempts have also been made to conduct a meta-analysis on the data of 18 usability evaluation studies [7].

Meta-analyses provide a systematic approach to assess a body of accumulating empirical results. However as Hartson et al. [7] experienced it is not always possible to conduct a meta-analysis as published studies sometimes lack to report the required elementary descriptive statistical data such as the standard deviation, sample size, and the mean. Fortunately we had access to all of the data from the six experiments and were therefore able to conduct an extensive meta-analysis. Although the sample size of 12 component tests might seem modest, the size of the effectiveness improvement brought about by component-specific measures was so large that it already stood out significantly in this relative small-scale meta-analysis. Before presenting the results of this meta-analysis, a brief overview is given of the six experiments.

A. Description of Experiments

1) Fictitious system

The first experiment in the list is an explorative experiment [29]. It was a first attempt to record the interaction on multiple layers and study the components-specific efficiency measure. In this experiment 80 university students (54 male and 26 female) between the age of 17 and 27 years old (M = 21.74, SD = 2.30) operated a fictitious user interface, which presented an abstraction of a mode-selection dialogue that is essential in multi-layered user interfaces. In a training session that preceded the task, the users received one out of eight instruction sets, which were created by providing or withholding information about three components (Selector, Map, Rotator) of the system. The interface consisted out of six symbols, such as symbols of music instruments, type of fruit, and transportation devices. A combination of clicking on the symbols of two transportation devices allowed the participant to rotate the three symbols of the music instruments. A fourth fruit symbol indicated with music instrument was selected for rotation. In the experiment participants were asked to rotate a specific music instrument. The experiment showed that users' knowledge about a component affected the number of messages it received. Because of the abstract nature of the user interface, users were not asked to rate the perceived ease of use or their satisfaction when operating the components.

2) Mobile Telephone

The second experiment [8] was conducted to validate the component-specific measures by using a PC emulator of mobile telephone. The emulator allowed users to make a call, check their voice mail, send and receive short text messages, and read and edit an address list and a diary. Eight different versions of the mobile telephone were constructed by manipulating three components that were responsible for the way users could activate functions in the telephone (Function Selector), input alphabetic characters (Keypad), and send text messages (Send Text Message). One version of these three components was always relatively easier to use, while the other version was designed to be more difficult to use. All usability variations addressed the complexity of the dialogue structure of the components, which could be understood in terms of the Cognitive Complexity Theory [14]. All 80 participating users (53 males and 27 females) were university students between the age of 18 and 28 years old (M = 21.43, SD = 2.27), and they were randomly assigned to one of the eight mobile telephone emulators. In the experiment participants were asked to make a call, send a short text message, and to add a person to the address list.

3) Room Thermostat

The third, fourth and fifth experiments [17] were part of a series of experiments to study the effect inconsistency between components may have on the usability of individual components. The results of these experiments showed that overall usability of an entire interactive system cannot always be predicted solely by looking at the individual usability of its components. All three experiments were conducted simultaneously. The 48 university students (32 males and 16 females) that participated had an age between 18 and 27 years old (M =21.69, SD = 2.03). They were asked to operate a version of a room thermostat, a web enabled TV set, and a microwave or a radio alarm clock.

The experiment with the room thermostat focused on the effect inconsistency could have on components that operated within the same interaction layer. Four PC emulators of a room thermostat were constructed which allowed users to set the daytime temperature and the nighttime temperature. Manipulating the two components responsible for setting these two temperatures resulted in these four room thermostat versions. In one version the control had a display with a moving pointer and a fixed scale, in the other version the display had a fixed pointer and a moving scale. In the experiment, participants were asked to set both the daytime and nighttime temperature.

4) WebTV

The effect of inconsistency between components operating on different layers was studied with four PC emulators of a web enabled TV-set including a remote control. These four emulators were constructed by manipulating the browser and the layout of the web pages. One version of the browser allowed both horizontal and vertical movement of the selection cursor, while the other version only allowed horizontal movement with the selection cursor jumping vertically only at the edges of a page. In the experiment the users were asked to find the web page that gave the departure times of a specific bus. The bus web site had two kinds of layout. One layout, the matrix layout, placed the web links in a web page both on the same line and one below the other. The other layout, the list layout, placed all links one below the other. Besides the browsing functionality, the TV set allowed users to switch the TV on and off, select TV channels, and change the volume.

5) Microwave & Radio Alarm Clock

The fifth experiment looked at the effect of inconsistency between the application domain and implementation of a particular component. The experiment tested a timer component in a microwave and in a radio alarm clock. In the radio alarm clock, the timer determined when the radio should be switched on, and in the microwave, the timer determined when cooking should start. Two versions of the timer were developed. The only difference between the two versions was the symbol they used to indicate that the timer was showing the time the timer would go off and not the normal time. In one version, the symbol was a ringing mechanical alarm clock; in the other version, the symbol was a hot dish. Whereas the microwave allowed users to start or stop the microwave and to set the clock, the cooking period, and the power, the radio alarm clock allowed users to set the clock, the radio channel, the volume, and switch the radio on or off. Participants that were randomly assigned to use the microwave, where asked to set the timer, the cooking time and the power. Participants assigned to use the radio alarm clock were asked to set the alarm time, the radio channel and the volume.

6) Calculator

The last experiment [20] studied the effect memory demand could have in linking the usability of two individual components together. Again, this experiment showed the problems in predicting the usability of the entire system solely on the usability of the individual components. From all experiments presented here, this is the only experiment with a within-subjects design. The 24 university students (16 males and 8 females), age between 19 and 25 years (M = 21.33, SD = 2.16), that participated had to solve equations, with different degrees of complexity, with two calculators. The calculators had two interaction components, the editor component, responsible for establishing an equation, and the processor component, responsible for processing the equations and if requested storing the results in one of the 6 memory places. Although both calculators had the same processor component, they were implemented with different editor components. One editor version had a small display, only capable of showing a small part of an equation, while the other version was equipped with a large display, allowing users to see the entire equation.

B. Meta Analysis

The first step of the meta-analysis was reanalysing the data from these studies. Sixty ANOVAs were conducted both on the overall measures (keystrokes, overall ease-of-use, and overall satisfaction) and component-specific measures (number of messages, component-specific ease-of-use and component-specific satisfaction) collected in the experiments. The ANOVAs gave the probability that the difference between the versions of a component had happened by chance. Table VI shows the results of 24 ANOVAs done on the number of messages (left side of the table) and keystrokes (right side of the table) measures. Table VII shows the results of the 18 ANOVAs done on the component-specific ease-of-use measures and the overall ease-of-use measures. And finally, Table VIII shows the results of the 18 ANOVAs done on the component-specific satisfaction and overall satisfaction measures.

The next step was to study whether, on average, component-specific measures are statistically more powerful than overall measures when comparing component versions. Power in this case is an expression of the likelihood of detecting a significant difference, i.e. a p-value < 0.05, if there is one. A look at the tables shows that on average the F-ratios obtained for the component-specific measures (16.80) is larger than the F-ratios based on overall measures, which is 10.65 (Table III). As mentioned before, the F-ratio is also affected by the sample size. A more pure measure and therefore used traditionally in meta-analyses is the partial effect size (η_p^2). This measure relates to the strength of the measurement and does not change if the sample size is increased or decreased in a test. Partial η^2 is the proportion of the total variation that is attributable to the difference between the versions of a component, and is defined as:

$$\eta_p^2 = SS_b / \left(SS_b + SS_w\right). \qquad (4)$$

For each of the 60 ANOVAs, Tables VI- VIII also show the partial η^2. Overall component-specific measure seems significantly more powerful than the overall measures. All the partial η^2 of the ANOVAs on the component-specific measures were simultaneous larger than the partial η^2 of the ANOVAs on the overall measure for five of the nine components that were measures with all three component-specific measures. The probability that this would happened for a single component by random chance is 1/8, i.e. $(1/2)^3$ assuming that by random chance alone the partial η^2 component-specific measure had one in two chance of being larger than its overall counterpart. A simple binomial test shows that the chance that at least five out of nine of these cases happened by random chance alone is

$$p = \sum_{r=5}^{9} \binom{9}{r} \left(\tfrac{1}{8}\right)^r \left(\tfrac{7}{8}\right)^{9-r} = 0.002\,, \qquad (5)$$

which is well below the often use 0.05 threshold value. In addition it is also possible to consider the size of the improvement made by the three component-specific measures. Table III shows that the average partial η^2 for the component-specific measures was 0.202 and 0.122 for the overall measures. However, how representative is this observed difference, or in other words what is the likelihood that it was simply caused by sampling error? To study this possibility, three ANOVAs were conducted on the partial η^2-s obtained in the 60 ANOVAs, hence the name meta-analysis. The first ANOVA compared the partial η^2-s from the keystrokes measures with the partial η^2-s from the messages measures (Table VI). Because the data is taken from the same statistical F-test, only with different measures, individual difference can be cancelled out by only looking how the effect size of a test differs from its individual mean ($\overline{X}_{i.}$) in each component test (6).

$$X'_{ij} = X_{ij} - \overline{X}_{i.} \qquad (6)$$

For example, the partial effect size of 0.109 for the F-test on the number of messages received by the Selector component and the partial effect size of 0.063 for the F-test on the number of keystrokes (Table VII), would for the meta analysis be transformed to 0.109 - (0.109 + 0.063) / 2 = 0.023 and 0.063 - (0.109 + 0.063) / 2 = -0.023. In other words, the meta-analysis was a within-subjects analysis or an ANOVA with *repeated-measures.* Note also that for a within-subjects analysis, df_w is defined as (n – 1)(k – 1), which means that df_w is (12 – 1)(2 – 1) = 11. The first row of Table IV shows the results of the ANOVA. As expected the p-value, with a value smaller than 0.05, indicates a significant difference. The mean partial η^2 for tests based on the number of messages received was 0.183, whereas on the number of keystrokes 0.081. Although the absolute values are of less interest here, the difference of 0.102 show the improvement the number of messages measure can make. Putting this into perspective, Cohen [40] characterises a small effect as 0.01, a medium effect as 0.06 and a large effect as 0.14. Therefore, this improvement suggests that a test based on an overall measure with a medium effect size can be improved into a test with a large effect size by using component-specific measures.

TABLE III
MEAN F-RATIO AND PARTIAL ETA SQUARED OF THE 60 ANOVAS ON COMPONENT-SPECIFIC MEASURES AND OVERALL MEASURES

	Component-specific		*Overall*	
	$\overline{F}$	$\overline{\eta}_p^2$	$\overline{F}$	$\overline{\eta}_p^2$
Object. perform.	16.54	0.183	6.15	0.081
Ease-of-use	16.51	0.209	13.37	0.151
Satisfaction	17.35	0.213	12.43	0.136
Total	16.80	0.202	10.65	0.122

TABLE IV
RESULTS ANOVAS ON PARTIAL ETA SQUARED

	df_b	df_w	SS_b	SS_w	F	$p.$
Object. perform.	1	11	0.0634	0.0600	11.64	0.006
Ease-of-use	1	8	0.0151	0.0358	3.39	0.103
Satisfaction	1	8	0.0266	0.0363	5.83	0.042

The overall measure in the case of the objective efficiency measure was the number of keystrokes made. The reason for taking this measure, as an overall objective efficiency measure instead of e.g. task time, was that in some cases differences existed between optimal task performances when executing a task with different versions of a component. Both the number of messages and keystrokes can easily be corrected for this by subtracting the a-priori differences, which is not directly possible for other objective measures. Still, the corrected keystrokes measure seems an appropriate indicator of the power of overall measures because of the reported high correlation with other measure such as the time to complete a task [8].

Another ANOVA with repeated measures was conducted on the partial η^2 values of the tests that were based on the ease-of-use measure (Table VII). Although the mean partial η^2 value of the component-specific measure (0.209) was larger than that of overall measure (0.151), this difference was not significant as Table IV shows. Finally, a similar ANOVA conducted on the satisfaction measure (Table VIII), did again reveal a significant difference. The mean partial η^2 value of the component-specific measure (0.213) was again larger than that of the overall measure (0.136), showing an average improvement of 0.077. This estimation however could be too conservative as a consequence of the experimental set-up of all these experiments where users received both component-specific and overall questions at the same time. The recollection triggered by the component-specific questions might have influenced the users when answering the overall questions. However, when using component-specific questions, this approach of including overall questions in the questionnaire seems realistic, as testers might have a tendency to ask rather too much than too little.

Although component-specific questions where taken from standard questionnaires, originally developed as overall measures, they seem also reliable measures for components-specific measuring. For psychometrics instruments, such as this questionnaire, Cronbach's alpha (7) is a frequently used reliability measure, which gives an indication of the extended that a set of questionnaire items measures the same latent variable. Cronbach's alpha is defined as

$$\frac{N \times \bar{r}}{\left(1+(N-1)\times \bar{r}\right)}. \tag{7}$$

Whereby N is the number of questionnaire items and $\bar{r}$ is the average of all Pearson correlations between the questionnaire items. Table V shows that both the ease-of-use and satisfaction questions had an acceptable reliability of 0.7 - 0.8 or more than this minimal level often recommended [41]. Note that the data of the calculator experiment had to be restructured for this analysis. As mentioned before, this experiment had a within-subjects design. The data was therefore split into two groups to create a between-subject design and afterwards the Cronbach's alpha was calculated. The high Cronbach's alpha values indicate consistent results across the different questions. Or in other words, the six ease-of-use questions and the two satisfaction questions were measuring the same underlying construct, i.e. the ease-of-use or satisfaction of a component or system.

TABLE V
CRONBACH'S ALPHA DERIVED FROM RELIABILITY ANALYSES

Application / Component	Ease of Use	Satisfaction
Mobile Telephone	**0.85**	**0.90**
Function Selector	0.87	0.75
Keypad	0.85	0.86
Send Text Message	0.89	0.81
Room thermostat	**0.82**	**0.91**
Daytime temperature	0.92	0.91
Night time temperature	0.92	0.94
WebTV	**0.91**	**0.92**
Browser	0.90	0.90
Web Pages	0.89	0.86
Microwave & Radio alarm clock	**0.94**	**0.84**
Timer	0.92	0.92
Calculator	**0.96**	**0.94**
Editor	0.97	0.96

V. DISCUSSION

To summarize the findings of the meta-analysis, the component-specific measures were on average statistically more powerful than their overall counterparts when comparing component versions. Detailed examination found that this was the case for the component-specific efficiency measure and the subjective satisfaction measures. Although failing to reach a significant level, the component-specific subjective easy-of-use measure also points in the direction of improved statistical power.

The analysis, like any, also has its limitations. For instance, because of the relative newness of the testing method we could only base the meta-analysis on the results of our own experiments, and not yet of that of others. Also all participants were university students. Generalising these results to other group of the general population should therefore be done with caution. Future meta-analyses, therefore, will be needed to see whether others can replicate these findings in another context. This will also help to improve the estimation of the effect component-specific testing has on the effect size. Still the presented effect sizes allow testers to plan their test strategy. They can set the statistical power they want their test to have, conduct a-priori power analyses, calculate the number of users needed, and compare this with the effort involved of applying an overall or the component-based testing method.

For example, consider an experiment in which testers would want to examine the objective efficiency of two versions of a component, and they would want a 60% chance of finding a possible significant difference ($p. < 0.05$). Running a-prior power analysis with GPower[2] [42] based on the average partial η^2 found in the first row in Table III would indicate that at least 24 users would be needed when using the component-specific measure and 58 users when using the overall measure. Now it is up to the testers to compare this 59% reduction in the number of users with the possible extra effort involved in obtaining data for the component-specific measure.

A. Limitations of the Testing Method

The power of component-specific measures is based on the idea that usability problems are contained mainly to the interaction of a single component. However, factors such as inconsistency [17] or mental load [20] can make that usability problem spread across the interaction with other components. In these cases, overall measures might be more effective as they are perceptive to all usability problems in the interaction. This suggests therefore a test strategy of not only selecting overall measure or only component-specific measures, but a strategy in which both types of measures are collected. With this strategy, engineers benefit from the statistical power of component-specific measures, and are still able to cope with situations where usability problems go beyond a single component. The extra effort of collecting the overall measures, such keystrokes or overall usability questions, might be small once engineers have at least a prototype application in which different versions of the component can be embedded. Besides including recording instructions in the component, keystroke data could be obtained by including recording instructions into components operating on the lowest-level layer or alternatively by using an external software logging tools that is able to record events sent to an application on an operation system level.

The results of the objective component-specific efficiency measure can only be understood in the context a specific component. The assumption is that users spent a similar amount of effort when sending a user message to the different versions of a component, which might not be the case for messages sent to other components. In other words, the difference in physical event effort represented by a difference of 10 messages received between two versions of the Play List component might not be the same as the effort represented by a difference of 10 messages received between two versions of the Player component from CD-player example. The assumption of similar amount of effort in creating message seems reasonable for high-level components as they rely on the same lower-level layer to mediate the message exchange. However, for component operating in the lowest-level layer, a cycle of the control loop can involve different amounts of effort. Take for example the evaluation of the Sam text editor by Thomas [43]. He tried to compare the relative command frequencies of the Sam text editor to other systems reported in the literature. The Sam's logging system recorded low-level mouse actions, like mouse clicks and positioning whereas the Unix applications reported in the literature recorded actions on a higher-level of abstraction (e.g. complete command lines). A possible way to solve the problem, of variation in the effort to create a message, is the introduction of weighting factors for the messages to represent the differences in effort. This approach is also used in SVTP when, instead of different versions of a single component, different components in a system are compared with each other [15].

Except for self-made components or open source development, testers might not always have access to the messages exchange. Fortunately some effort is being made to address this. Software tools such as iGuess [44] are able to automatically insert recording code into Java applications without any need for access to the source code.

Another limitation of the study is that it mainly looks at the usability from a user perspective, and mainly ignores the developers' perspective. To be usable a component should also be easy to re-use, e.g. easy maintainable and modifiable, and developers should understand easily how a component interfaces with other components. Future research could consider possible empirical methods to evaluate these developers' usability aspects of a component. Combined with data from a user perspective this would establish a truly overall understanding of the usability of a component.

B. Other Empirical Evaluation Methods

Unit testing allows testers to focus on a specific part of the system, while using overall measures. Although this approach can be used for lower-level components, by asking users to complete elementary tasks, applying unit testing to test higher-level components is less effective as the task would also include interaction with mediating lower-level components and might consequently increase SS_w.

Existing Sequential Data Analysis (SDA) techniques [45] on recorded keystrokes allow testers to overcome the unit test limitation of using elementary instead of normal every day tasks in a test. These SDA techniques often pre-process the input data and filter out non-relevant input. Again these techniques are effective for lower-level, but not for higher-level components. Instead of recording the higher-level message exchange directly, these techniques attempt to generate the higher-level interaction from the recorded lower-level input, without taking into account the component's response and state when processing these messages. An indirect way of solving this would be to record the system state together with the user events and to envision the response of the system; see the work of Lecerof and Paternò [46] for an example.

Other usability evaluation methods, besides event-based usability testing, are often used to study applications, such as: Thinking-Aloud, Cognitive Walkthrough, and Heuristic Evaluations. These methods may in some cases be quicker to

[2] Note that GPower uses f instead of partial η^2. However f is defined as $f = \sqrt{\eta_p^2 / (1 - \eta_p^2)}$

come up with results, still they suffer from a substantial evaluator effect in that multiple evaluators or even test teams end up with different conclusions when testing the same application [47], [48]. Using theoretical concepts, such as control loops and layered interaction, combined with a set of related measures might reduce this evaluator effect.

C. Exploitation of the Testing Method

For user-centred design techniques to be effective they need to be aligned with the software development life cycle and it has to be clear where and how they should be used [49]. To answer the first part of this question, MVTP component-based testing methods can be used in two concurrent processes, namely: the *create* and the *deployment* process of the component-based engineering approach. The first process is responsible for the design and creation of new software components. Here engineers could compare different versions of a component and ship off the most usable version to a component library. Testing in this process is an efficiency step because it would affect many applications at once; usability problems related to the individual nature of the components are already eliminated before components are deployed in applications. Testing the components may, however, require development of at least a potential prototype application, as an actual application might not be available when developing a generic component library. The type of components that can be tested, range from very simple two states interaction component, such as a Sound On-Off indicator, to very complex interaction components, such as a drawing component in a word processor application. The important requirement however remains that these components have a state that a user can perceive and change.

In the deployment process, where components are used to create a new application, the role of the testing method is to help engineers to select the most usable component for their application from a set of off-the-shelf components that provide the same functionality. This test helps testers to understand how a component would function in the context of other components, a specific task, and a specific user group, which were probably unknown in the create process. In general, engineers can apply component-based testing in an assembly-project of any interactive applications such as PC applications or consumer electronics. Still because the component-specific efficiency measure assumes some toleration for user deviation in the task executing, systems such as password verification might be less suitable.

VI. Conclusions and Final Remarks

Component-specific measures were on average statistically more powerful than the overall usability measures when it came to comparing the usability of different versions of a component in the 12 component tests that were examined in the meta-analysis. Therefore, the testing method seems promising as a method suitable for engineers that apply a component-based software engineering approach. However, the real benefit will only become apparent when actual software engineers put it into practice and the usability of the final product is also assessed [50]. It will then become visible how much extra effort and money is involved and how it fits in with normal engineering routines. The development of a software tool that supports the testing method might help here. The tool should be an integrated part of the software engineers' development environment in order to make it more accessible and aligned with their work [49].

Another research direction is to adapt the testing method to be used outside the laboratory. This would require re-examination of the component-specific objective efficiency measure, because now the tester sets the users' goal, which would be inappropriate in normal field tests. Also remote capturing of the message exchange would allow for large-scale testing. Engineers could make different versions of a new component online available, which users could download in their application, such as different versions of a dictionary for a word processor application. Once the users start using the component, component-specific usability data could be collected over the network. This would provide engineers with data about the component use in the actual usage context.

APPENDIX

TABLE VI
RESULTS OF ANOVA ON OBJECTIVE EFFICIENCY MEASURES

Application /			Test Based on Component-Specific Measure					Test Based on Overall Measure				
Component	df_b	df_w	SS_b	SS_w	F	$p.$	η_p^2	SS_b	SS_w	F	$p.$	η_p^2
Fictitious system												
Selector	1	78	1872	15307	9.54	0.003	0.109	20034	299402	5.22	0.025	0.063
Map	1	78	4977	14560	26.66	<0.001	0.255	55862	263574	16.53	<0.001	0.175
Rotator	1	78	316	19221	1.28	0.261	0.016	15125	304311	3.88	0.053	0.047
Mobile Telephone												
Function Selector	1	78	379226	392002	75.46	<0.001	0.492	542851	2309387	18.34	<0.001	0.190
Keypad	1	78	25347	164567	12.01	0.001	0.133	259009	2593229	7.79	0.007	0.091
Send Text Message	1	78	9680	27221	27.74	<0.001	0.262	26975	982252	2.14	0.147	0.027
Room thermostat												
Daytime temperature	1	46	111	518	9.86	0.003	0.177	46	1523	1.39	0.245	0.029
Night time temperature	1	46	85	497	7.91	0.007	0.147	78	1496	2.38	0.130	0.049
WebTV												
Browser	1	46	2837	11136	11.72	0.001	0.203	3104	11264	12.68	0.001	0.216
Web Pages	1	46	234	1275	8.45	0.006	0.155	675	13693	2.27	0.139	0.047
Microwave & Radio alarm clock												
Timer[a]	1	46	83	3589	1.06	0.309	0.023	1170	64639	0.83	0.366	0.018
Calculator												
Processor × equation difficulty[b]	1	23	0.288	0.972	6.81	0.016	0.228	0.015	1.005	0.34	0.567	0.014

[a]The component-specific measure only counted the number of change-mode messages. [b]These results show the two-way interaction effect of the Editor version and Equation difficulty. The component-specific measure was based on the log transformation of the number of store requests sent to the processor.

TABLE VII
RESULTS OF ANOVA ON EASE-OF-USE QUESTIONS

Application /			Test Based on Component-Specific Measure					Test based on Overall Measure				
Component	df_b	df_w	SS_b	SS_w	F	$p.$	η_p^2	SS_b	SS_w	F	$p.$	η_p^2
Mobile Telephone												
Function Selector	1	78	24.4	80.9	23.51	<0.001	0.232	11.3	76.6	11.45	0.001	0.128
Keypad	1	78	9.7	78.4	9.63	0.003	0.110	1.0	86.9	0.91	0.343	0.012
Send Text Message	1	78	0.5	110.2	0.34	0.564	0.004	1.1	86.8	0.98	0.326	0.012
Room thermostat												
Daytime temperature	1	46	13.2	56.4	10.77	0.002	0.190	1.4	29.4	2.18	0.147	0.045
Night time temperature	1	46	18.3	49.4	17.08	<0.001	0.271	1.1	29.7	1.66	0.204	0.035
WebTV												
Browser	1	46	4.8	40.3	5.47	0.024	0.106	8.2	43.4	8.69	0.005	0.159
Web Pages	1	46	2.9	32.9	4.084	0.049	0.082	2.8	48.8	2.597	0.114	0.053
Microwave & Radio alarm clock												
Timer	1	46	7.8	58.1	6.16	0.017	0.118	5.9	45.0	6.04	0.018	0.116
Calculator												
Editor	1	22	56.2	17.3	71.51	<0.001	0.765	54.0	13.8	85.86	<0.001	0.796

TABLE VIII
RESULTS OF ANOVA ON SATISFACTION QUESTIONS

Application /			Test Based on Component-Specific Measure					Test Based on Overall Measure				
Component	df_b	df_w	SS_b	SS_w	F	$p.$	η_p^2	SS_b	SS_w	F	$p.$	η_p^2
Mobile Telephone												
Function Selector	1	78	24.2	119.9	15.75	<0.001	0.168	6.3	135.0	3.66	0.060	0.045
Keypad	1	78	14.9	149.9	7.74	0.007	0.090	2.6	138.7	1.48	0.228	0.019
Send Text Message	1	78	0.5	146.3	0.24	0.626	0.003	1.4	140.0	0.77	0.384	0.010
Room thermostat												
Daytime temperature	1	46	39.4	125.0	14.50	<0.001	0.240	3.5	111.0	1.46	0.233	0.031
Night time temperature	1	46	36.8	102.9	16.43	<0.001	0.263	3.0	111.5	1.238	0.272	0.026
WebTV												
Browser	1	46	20.0	85.5	10.78	0.002	0.190	15.76	79.7	9.09	0.004	0.165
Web Pages	1	46	9.2	73.8	5.73	0.021	0.111	4.4	91.1	2.21	0.144	0.046
Microwave & Radio alarm clock												
Timer	1	46	7.1	112.8	2.91	0.095	0.059	8.3	93.0	4.12	0.048	0.082
Calculator												
Editor	1	22	103.5	27.8	82.05	<0.001	0.790	93.3	23.4	87.84	<0.001	0.800

TABLE IX
GLOSSARY OF SYMBOLS

Symbol	Description	Equation
df_b	Degrees of freedom between groups	
df_w	Degrees of freedom within groups	
SS_b	Sum of Squares between groups	(2)
SS_w	Sum of Squares within groups	(3)
F	Ratio of between-sample and within-sample variation	(1)
$p.$	Probability that, under the assumption of equal group means, an observed difference between group means occurred by random chance alone.	
η_p^2	Partial effect size	(4)

REFERENCES

[1] B. M. Horowitz and J. H. Lambert, “Assembling off-the-self components: “Lean as you go” system engineering,” *IEEE Trans. Syst., Man, Cybern. Part A*, vol. 36, pp. 286-297, March 2006.

[2] W. Bewley, T. L. Roberts, D. Schroit, and W. L. Verplank, “Human factors testing in the design of Xeror’s 8010 Start Office workstation,” in *Proc. of CHI*, 1983, pp. 72-77.

[3] S. Rosenbaum, J. A. Rohn and J. Humburg, “A toolkit for strategic usability: Results from workshops, panels, and surveys,” in *Proc. CHI’00*, 2000, pp. 337-344.

[4] I. Bark, A. Følstad and J. Gulliksen, “Use and usefulness of HCI methods: Results from an exploratory study among Nordic hci practitioners,” in *Proc HCI’05*, 2005, pp. 201-217.

[5] Y. G. Ji and M. H. Yun, “Enhancing the minority discipline in the IT industry: A survey of usability and user-centered design practice,” *Int. J. of Hum. Comp. Inter.*, vol. 20, pp. 117-134, 2006.

[6] J. Nielsen and R. Molich, “Heuristic evaluation of user interfaces,” in *Proc. CHI’90*, 1990, pp. 249-256.

[7] H. R. Hartson, T. S. Andre and R. C. Williges, “Criteria for evaluating usability evaluation methods,” *Int. J. of Hum. Comp. Inter.*, vol. 13, pp. 373-410, 2001.

[8] W. -P. Brinkman, R. Haakma, and D. G. Bouwhuis, “Empirical usability testing in a component-based environment: Improving test efficiency with component-specific usability measures,” in *Proc. of EHCI-DSVIS*, LNCS 3425, 2005, pp. 20-37.

[9] W. -P. Brinkman, R. Haakma, and D. G. Bouwhuis, “Usability evaluation of component-based user interfaces,” in *Proc. of IFIP INTERACT*, 2001, pp.767-768.

[10] E. Chang and T. S. Dillon, “A usability-evaluation metric based on a soft-computing approach,” *IEEE Trans. Syst., Man, Cybern. Part A*, vol. 36, pp. 356-372, March 2006.

[11] M. Matera, M. F. Costabile, F. Garzotto and P. Paolini, “SUE inspection: An effective method for systematic usability evaluation of hypermedia,” *IEEE Trans. Syst., Man, Cybern. Part A*, vol. 32, pp. 93-103, Jan 2002.

[12] P. G Polson, C. Lewis, J. Rieman and C. Wharton, “Cognitive walkthroughs: A method for theory-based evaluation of user interfaces,” *Int. J. of Man-Mach. Stud.*, vol 36, pp.741-773, 1992.

[13] S. K. Card, T. P. Moran and A. Newell, *The psychology of human-computer interaction*. London: Lawrence Erlbaum, 1983.

[14] D. Kieras, and P. G. Polson, “An approach to the formal analysis of user complexity,” *Int. J. of Man-Machine Studies*, vol. 22, p. 365-394, 1985.

[15] W. -P. Brinkman, R. Haakma, and D. G. Bouwhuis, “Towards an empirical method of efficiency testing of system parts: a methodological study,” *Interacting with Comp.*, vol. 19, pp. 342-356, May 2007.

[16] M. Hertzum, “Component-based design may degrade system usability: Consequences of software re-use,” in *Proc OZCHI’00*, 2000, pp. 88-94.

[17] W. -P. Brinkman, R. Haakma, and D. G. Bouwhuis, “Consistency: a factor that links the usability of individual interaction components together,” in *Proc. ECCE-12*, 2004, pp.57-64.

[18] M. M. Taylor, P.S.E. Farrell and J. G. Hollands, “Perceptual control and layered protocols in interface design: II. The general protocol grammar,” *Int. J. Hum. Comp. Stud.*, vol. 50, 1999, pp. 521-555.

[19] R. Haakma, “Towards explaining the behaviour of novice users,” *Int. J. Hum. Comp. Stud.*, vol. 50, 1999, pp.557-570.

[20] W. -P. Brinkman, R. Haakma, and D. G. Bouwhuis, “Memory load: a factor that links the usability of individual interaction components together,” in *Proc. of HCI 2004*, vol. 2, 2004, pp.165-168.

[21] B. E. John, L. Bass, M. -I. Sanchez-Sequra adn R. J Adams, "Bringing usability concerns to design of software architecture," in *Proc. of EHCI-DSVIS*, LNCS 3425, 2005, pp. 1-19.

[22] C. Gram and G. Cockton, *Design principles for interactive software*. London: Chapman & Hall, 1996.

[23] G. E. Krasner, and S. T. Pope, “A cookbook for using the Model-View-Controller user interface paradigm in Smalltask-80,” *J. of objected-oriented programming*, vol. 1, pp. 27-49, Aug./Sept.1988.

[24] J. Coutaz, “PAC, an object oriented model for dialog design,” in *Proc. of IFIP INTERACT*, 1987, pp. 431-436.

[25] F. Paternò, *Model-based design and evaluation of interactive applications*. London: Springer, 2000.

[26] C. S. Carver, and M. F. Scheier, *On the self-regulation of behavior*. New York: Cambridge Univ. Press, 1998.

[27] W. T. Powers, *Behavior: The control of perception*. Chicago: Aldine Publishing Company. 1973.

[28] P. S. E. Farrell, J. G. Hollands, M. M. Taylor, and H. D. Gamble, “Perceptual control and layered protocols in interface design: I. Fundamental concepts,” *Int. J. of Human-Computer Studies*, vol. 50, pp. 489-520, Jun. 1999.

[29] W. -P. Brinkman, “Is usability compositional?,” Ph.D. thesis, Technische Univ. Eindhoven, Eindhoven, The Netherlands, 2003.

[30] D. C. Howell, *Statistical methods for psychology*, 5th ed. London: Duxbury, 2001.

[31] W. D. Coleman, R. C. Williges, and D. R. Wixon, “Collecting detailed user evaluations of software interfaces,” in *Proc. of the Human Factors Society - 29th Annual Meeting*, 1985, pp. 240-244.

[32] W. -P. Brinkman, R. Haakma, and D. G. Bouwhuis, “The theoretical foundation and validity of a component-based usability questionnaire,” *Beh. and Inf. Techn.*, to be published.

[33] F. D. Davis, “Perceived usefulness, perceived ease of use, and user acceptance of information technology,” *MIS Quarterly*, vol. 13, pp. 319-340, Sep.1989.

[34] W. J. Doll, A. Hendrickson, and X. Deng, “Using Davis’s perceived usefulness and ease-of-use instruments for decision making: A confirmatory and multigroup invariance analysis,” *Decision Sciences*, vol. 29, pp. 839–869, Fall 1998.

[35] J. R. Lewis, “IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use,” *Int. J. of Human-Computer Interaction*, vol. 7, pp. 57-78, 1995.

[36] G. J. Shaw, “User satisfaction in group support systems research: a meta-analysis of experimental results,” in *Proc. 31st System Sciences*, vol. 1, 1998, pp. 369-369.

[37] S. Hanque and S. Srinivasan, “A meta-analysis of the training effectiveness of virtual reality surgical simulators,” *IEEE Trans. Inf. Techn. Biomed.*, vol. 10, pp. 51-58, 2006.

[38] J. Miller, “Can results from software engineering experiments be safely combined?,” in *Proc. Softw. Metrics Symp.*, 1999, pp. 152-158.

[39] W. Hayes, “Research synthesis in software engineering: A case for meta-analysis,” in *Proc. Softw. Metrics Symp.*, 1999, pp. 143-151.

[40] J. Cohen, *Statistical power analysis for the behavioral sciences,* 2nd ed. New York: Academic Press, 1988.

[41] T. K. Landauer, “Behavioral research methods in human-computer interaction,” in *Handbook of human-computer interaction*, M. G. Helander, T. K. Landauer, and P. V. Prabhu, Eds. Amsterdam: Elsevier, 1997, pp. 203-227.

[42] E. Erdfelder, F. Faul, and A. Buchner, “GPOWER: A general power analysis program.,” *Beh. Research Meth., Instr., & Comp.*, vol. 28, pp. 1-11, 1996.

[43] R. C. Thomas, *Long term human-computer interaction: An exploratory perspective*, London: Springer, 1998.

[44] I. McLeod, H. Evans, P. Gray, and R. Mancy, "Instrumenting bytecode for the production of usage data," in *Proc.of CADUI*, 2005, pp.185-196.
[45] P. M Sanderson, and C. Fisher, "Exploratory sequential data analysis: qualitative and quantitative handling of continuous observational data," in *Handbook of human factors and ergonomics*, 2nd ed. G. Salvendy, Ed. Chichester: Wiley-Interscience, 1997, pp. 1471-1513.
[46] A. Lecerof, and F. Paternò, "Automatic support for usability evaluation," *IEEE Trans. Software Eng.*, vol. 24, pp. 863–888, Oct. 1998.
[47] M. Hertzum, and N. E. Jacobsen, "The evaluator effect: A chilling fact about usability evaluation methods," *Int. J. of Human-Computer Interaction*, vol. 13, pp. 421-443. 2001.
[48] R. Molich, M. R. Ede, K. Kaasgaard, and B. Baryukin, "Comparative usability evaluation," *Beh. and Inf. Techn.*, vol. 23, pp. 65-74. Jan. 2004.
[49] A. Seffah, and E. Metzker, "The obstacles and myths of usability and software engineering," *Communications of the ACM*, vol. 47, pp. 71-76, Dec. 2004.
[50] N. A. Stanton, and M. S. Young, "What price ergonomics?," *Nature*, vol. 399, pp. 197-198, May 20 1999.

Willem-Paul Brinkman received his BSc in information technology from the Hogeschool Eindhoven in 1995. In 1998 he received his MSc in technology & society at the Technische Universiteit Eindhoven, the Netherlands, where he also obtained his PhD degree in 2003.

Since 2007, he is an assistant professor in the Mediamatica Department at Delft University of Technology in The Netherlands. He is also a Lecturer at Brunel University in the UK. His research interests lie in the area of the human-computer interaction and e-learning.

Dr. Brinkman is currently a board member of the European Association of Cognitive Ergonomics (EACE), and a member of the programme committee of BCS HCI conference 2005-2008.

Reinder Haakma obtained his MSc in electrical engineering from University of Twente, the Netherlands in 1985, and his PhD degree from the Technische Universiteit Eindhoven, the Netherlands in 1998.

He is currently department head of Media Interaction department of the Digital Lifestyle Technology sector at Philips Research Laboratories Eindhoven, which he joined in 1986. From 1990 till 1995, he worked as a scientist at the Center for Research on User-System Interaction (IPO), Eindhoven. Between 1995 and 1997 he has been a visiting scientist at Philips Research Briarcliff, New York, USA. His research interests include usable and dependable systems.

Don G. Bouwhuis studied experimental psychology and mathematical psychology in 1968 at the University of Nijmegen, the Netherlands.

He joined the Institute for Perception Research (IPO), Eindhoven in 1968 where he was involved in product ergonomics and in product preference scaling models. Since 1988 he is a full professor of Technical Psychonomics at the Technische Universiteit Eindhoven at the Department of Technology Management. From 2001 he has become the scientific director of the J.F. Schouten School for User-System Interaction Research.

Prof. Bouwhuis serves as editor of the journal 'Gerontology', and as member of the editorial board of the journal Universal Access in the Information Society.