Preliminary version of: Brinkman, W.-P., Haakma, R., & Bouwhuis, D.G. (2007). Towards an empirical method of efficiency testing of system parts: a methodological study, *Interacting with Computers*, vol. 19, no. 3, pp. 342-356.

Towards an empirical method of efficiency testing of system parts: a methodological study

Abstract

Current usability evaluation methods are essentially holistic in nature. However, engineers that apply a component-based software engineering approach might also be interested in understanding the usability of individual parts of an interactive system. This paper examines the efficiency dimension of usability by describing a method, which engineers can use to test, empirically and objectively, the physical interaction effort to operate components in a single device. The method looks at low-level events, such as button clicks, and attributes the physical effort associated with these interaction events to individual components in the system. This forms the basis for engineers to prioritise their improvement effort. The paper discusses face validity, content validity, criterion validity, and construct validity of the method. The discussion is set within the context of four usability tests, in which 40 users participated to evaluate the efficiency of four different versions of a mobile phone. The results of the study show that the method can provide a valid estimation of the physical interaction event effort users made when interacting with a specific part of a device.

Keywords efficiency, usability testing; HCI methodology; usability evaluation method; log file analysis; empirical method.

1. Introduction

Evaluating the usability of a device on dimensions such as effectiveness, efficiency and satisfaction (ISO, 1998) has received considerable research attention in the past few decades. Researchers have come to realise that the acceptance of a system is significantly dependent on for example the ease with which people can operate a system (e.g. Davis, 1989; Davis and Venkatesh, 2000). Empirical usability evaluation methods such as user observations, questionnaires, and interviews are all tools that engineers can use to examine the usability of their system. Still, these methods do not provide quantitative data about the actual use of specific parts of the system, critical information when engineers apply a component-based software engineering (CBSE) approach. Because of the popularity of CBSE, it is important to have a suitable evaluation approach.

CBSE can be regarded as a response to the increase in the complexity of systems. As the complexity increases, design, development, and maintenance become more difficult. In response, software engineers have moved away from dealing with a system as a whole, and instead favour a more modularised or a component-based approach. The aim is to create autonomous components that hide the internal complexity from other components. This idea is considered as one of the major success factors behind object-oriented development; it reduces the complexity of large software projects and improves the maintainability and reliability of a system (Cox, 1990). In this approach, systems are not developed from scratch but are assembled by

using pre-produced parts (e.g. pop-menus, radio buttons, or more complex components such as a spell checker or an email component), which can be used in different applications. The promise of CBSE is reduced development cost and time, since ready-made and bespoke components can be used and re-used (Aykin, 1994).

CBSE and software development in general have also been studied in the context of interactive systems. For example, the IFIP's Working Group 2.7(13.4) (Gram and Cockton, 1996) has studied links between a set of user-perceivable properties of interactive systems, such as goal and task completeness, flexibility and robustness, with a set of software phenomena as seen from the software engineers' perspective, such as software architecture, tools, documents and code. They argue that CBSE can improve system modifiability and maintainability, which increases the system's lifetime and the ease of keeping it operational. Attempts have also been made (John et al., 2005) to develop usability-supporting architectural patterns, which address usability problems that arise because of software modularisation, such as responding to a user's cancellation command across a series of components. Furthermore, the compositional view has been used to explain and predict human-computer interaction. For example, Taylor (1988a) has proposed a layered interaction framework. He explained how users and components of a system interact across multiple layers. Several interaction mechanisms have been studied within this framework, such as a general protocol grammar (Taylor et al. 1999), diviplexing and multiplexing (Taylor and Waugh, 2000), communication synchronisation (Taylor, 1989), and layered feedback (Haakma, 1999). The framework has also been suggested (Haakma, 1998; Hilbert and Redmiles, 2000; Taylor, 1988b) as a framework for evaluating human-computer interaction, in other words, component-based usability evaluation. Usability can be considered a multi-dimensional construct (ISO, 1998). The evaluation method put forward in this paper, however, focuses only on a part of the efficiency dimension. It is therefore a first exploratory step towards the wider idea of component-based usability evaluation.

1.1 Usability and efficiency evaluation

Selection and customisation of components, when producing a new application, remains a key challenge in the CBSE approach. Applying this approach to interactive systems, gives efficiency evaluation a potential active role in the selection and customisation process of the components. Information about the efficiency of the different components in a new application would help to direct the software-engineers' attention towards components that decrease the overall efficiency of the application. Besides their ability to locate efficiency problems, the effectiveness of these methods also depends on their ability to accommodate a particular development approach, in this case CBSE. For example, simulation models such as GOMS (Card et al., 1983), or the Cognitive Walkthrough (Polson et al., 1992), that explain interaction from a cognitive model can be used when engineers start off by specifying the user interface and the user task. Without these specifications, but with a working prototype built from existing components, heuristic evaluation (e.g. Nielsen and Molich, 1990) or a user test (e.g. Rubin, 1994) would be effective. Heuristic evaluation, and new techniques, such as CASSM (Blandford et al., 2005), are analytical in nature. They do not directly analyse actual user behaviour or opinions. They are often employed to identify usability problems at an early stage of development when it is still relatively less expensive to make adjustments to the system. However, using off-the-shelf components shortens development time, making system adjustments less expensive.

CBSE could therefore make the more time consuming empirical oriented techniques more attractive. These techniques such as questionnaires, user observations in the lab or in the field, ground their findings in actual applications usage and not in consolidated knowledge from previous findings with other devices. Interpreting the data and relating this back to design suggestions can however be a difficult step. For example, when it comes to observation-based evaluation, such as a qualitative oriented usability test, evaluators need to provide this link. They observe the users interacting with the system, write down the problem users encounter, but then the evaluator has to attribute these problems to parts of the system so engineers can try to solve them. This process has been criticised as being very subjective, in that different evaluators or entire teams of evaluators come up with completely different lists of problems when examining the same system (Molich et al., 2004), or even when examining the same observation tapes (Hertzum and Jacobsen, 2001). Evaluators might therefore benefit from additional quantitative information about the usability, or more specifically the efficiency with which users operate individual parts of a system. Existing quantitative indicators, however, measure usability on an overall level. For example, the average task completion might be 5.2 minutes, or the overall satisfaction, learnability or mental load score on a scale from 1 to 7, might be 4.3. How this relates to a design suggestion to improve a particular part of the system is unclear. Some usability questionnaires therefore include questions on specific parts of the system, such as font size, error messages and help facilities (e.g. Chin et al., 1988). Unfortunately, when it comes to behavioural measures, a link to the efficiency of a specific part of the system is currently lacking.

1.2. Component evaluation strategies

When evaluating components of a system, there are two basic strategies that can be applied: stepwise testing, and big bang testing (Broekman and Notenboom, 2003). Stepwise testing means that the test starts with a single or a limited number of components and is extended with other components each time the test results are satisfactory. If at some point the test results indicate a problem, the components that last entered the test are seen as (partly) causing it. Instead of actually physically extending the system, the user task can be extended stepwise, while using the complete system. Although this strategy helps to identify components that are part of an efficiency problem, the evaluation is limited as users only perform a scaled down task, and some efficiency problems will only emerge when users are performing a full task, requiring interaction with a large number of components. Furthermore, this strategy does not provide developers with a list of the components' efficiency to prioritise the potential improvements. Instead, the developers have to make improvements to the tested components before other components can be included in the next test cycle.

When testers apply a big bang testing strategy, the whole system is tested at once. This approach is very appropriate for conducting summative evaluation; however, applying component-specific efficiency measures would make it possible to also obtain formative information, as it directs developers to components that need improvement. This might help to enhance existing evaluation methods, which have received criticism for their ineffectiveness in finding real problems that lead to changes in a new version of a system (John and Marks, 1997).

In this paper we therefore present a method that addresses a part of this gap in existing usability evaluation methods. It includes a component-specific objective

measure that is based on recorded interaction behaviour between the user and specific parts of the system. Although it provides evaluators with information that existing behavioural measures can not provide, it only covers a small part of the entire usability spectrum. The method looks at a part of the efficiency dimension by giving an indication about the physical interaction event effort that can be associated with the use of a component. Arguing for this evaluation method is not advocating the abandonment of existing evaluation methods but to enrich them with an extra evaluation instrument. Before describing the method, the following section gives a brief introduction into the compositional view. The aim will not be to review this large literature, or to deny their importance; neither will this section propose a new architecture or specification notation. Instead this section will simply try to define a component that could be evaluated with the proposed evaluation method. After describing the efficiency method with a detailed example, the validity of the method will be analysed. This section is followed by a discussion of the limitations of the method.

2. Introduction to the compositional view

Component-specific evaluation can be applied to interaction systems that employ software architectures that split the software into separate, but interacting parts —the components. Examples of these kinds of architectures are the Model-View-Controller (MVC) model (Krasner and Pope, 1988), the PAC (Presentation, Abstraction, Control) model (Coutaz, 1987), ICON (Input Configurator) (Dragicevic and Fekete, 2001) and in particular the York (Duke et al., 1993) and CNUCE agent model (Paternò, 2000). In these architectures, components interact with each other by sending messages, e.g. by making function or method calls, or assigning a value to properties of another component. With the user, the components interact by sending or receiving messages from the input and output devices such as displaying symbols on a screen or reading key presses from a keyboard. Fig. 1. illustrates a part of such compositional architecture for a web radio application as presented in Fig. 2. The system has a Station List component, responsible for selecting a radio station; a Receiver component, responsible for managing the internet connection with the radio station; and a Display component, responsible for showing feedback from other components. As Fig 1 shows, a component, such as the Receiver component, does not always receive its messages directly from the user, but sometimes from mediating components. The Receiver is therefore defined as operating on a higher-level layer than the mediating Station List component. This means that lowest-level layers make up the so-called “look and feel” of the system, as higher-level layers do not have direct physical interaction with the user. Thus in this context, interaction between users and the system extends the narrow system-centred view that interaction only relates to that part of the system that mediates (interfaces) between the user and the system —the user interface. The layered view includes all components that have a state, and exchange messages with the users, even if this is done through mediation of other lower-level components.

A similar layering principle has been suggested when it comes to cognitive processes (Carver and Scheier, 1998; Newell, 1990; Norman, 1984; Nielsen, 1986; Powers, 1973; Vallacher and Wegner, 1987). 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 finding

the right radio program. The layered interaction framework (Farrell et al., 1999; Taylor, 1988) combines these cognitive and system layering-principles, by aligning cognitive processes with system components into interaction layers. For example in Fig. 1, the Control Receiver process on the left is interacting with the Receiver component on the right. This interaction layer is regarded as a virtual interaction layer, since the message exchange is routed through a lower-level layer. Only at the lowest layer is the message exchange physical, e.g. key presses or symbols on the screen.

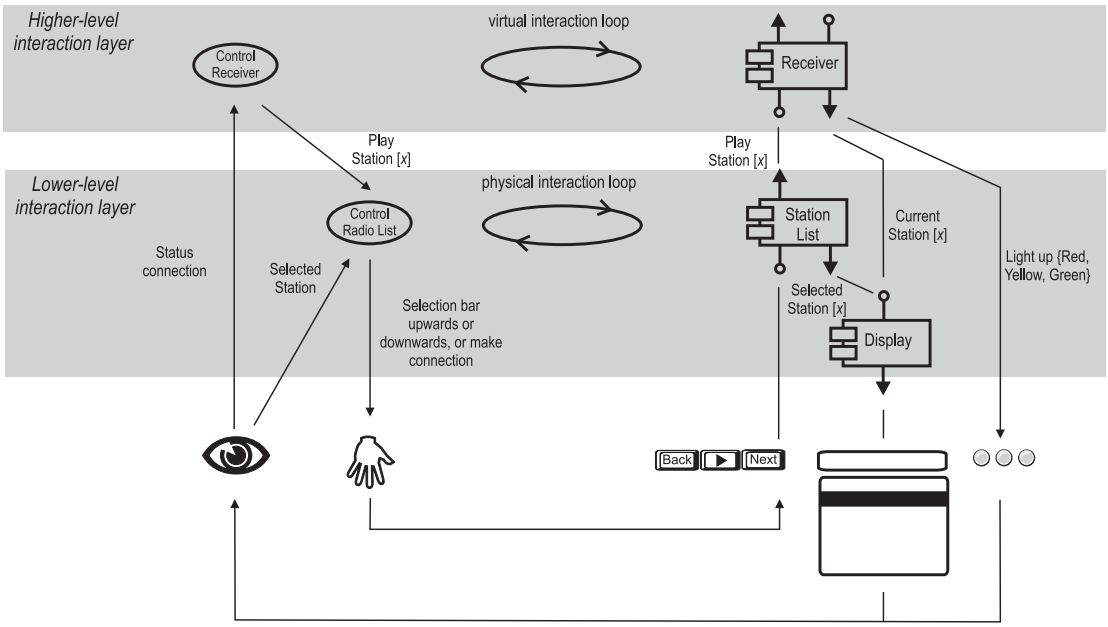

Fig. 1. Part of a layered interaction structure between a user, left, and a web radio on the right.

Another key concept of the framework is the notion of control loops. Users interact with components because they are not satisfied with the state they perceived the component to be in. In Fig. 2, for example, the selected station in the station list is BBC Asian Network, whereas users might want it to be BBC 7. Because of this mismatch, users start sending messages to the component, i.e. pressing buttons, until they receive feedback from the component that confirms that the selected station is BBC 7. Of course users might also stop sending messages if they give up on their desire, or they stop believing that sending messages will move the component forward to the desired state.

Norman (1984) talks about a single control loop between system and user, identifying four stages in a full cycle of the loop: forming the intention, selecting an action, executing the action, and evaluating the outcome. Farrell et al. (1988) take this one step further and talk about a control loop operating on each layer. With components that have their own state, which users can change and perceive or infer, each component can be placed in a control loop. This forms a key element for a potential behavioural measure, as it links user behaviour, pressing buttons, with the efficiency of controlling or regulating a component. For example, in an experiment (Brinkman, 2003) where one group of users was provided with prior information about the interpretation of some interface icons and another group not, a significant difference was found in the number of actions made by users of the two groups to complete the same task. The group without the prior information, needed more actions, or in other words, went through more cycles of the control loop to get the

system in the desired state. They had to explore the system to understand how to map their intentions with the right action, in this case clicking on the appropriate icons.

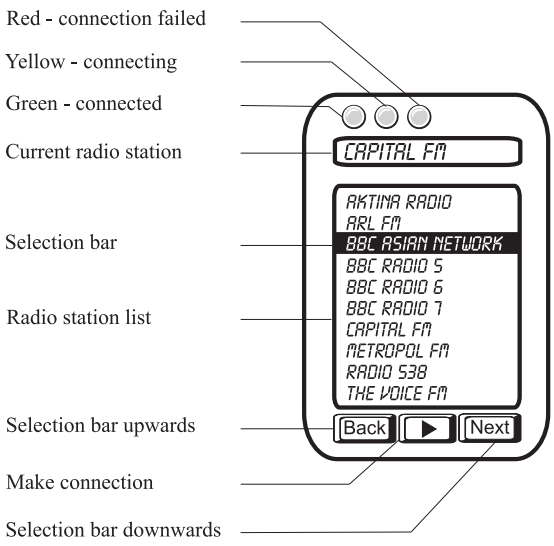

Fig. 2. Front of a web radio

With a link between interaction behaviour and software component established, it is now possible to move on to identify the kind of component that can be evaluated by examining the interaction behaviour. First of all, a component needs an independent state that users can change. Without one there is nothing to control. Secondly, components should provide the user with feedback about their state. Without feedback the user can not evaluate whether the desired state has been obtained. Software components that have these properties have been defined as *interaction components* (Brinkman et al., 2005b). The definition has clear parallels to that of interactors (Duke et al., 1993), however no reference is made to the component's internal organisation. In the example of the web radio, the Radio List, and the Receiver components are interaction components. The Display component on the other hand is not. Its state depends on the state of the other two components. Studying the user behaviour to understand the efficiency of the display can only be achieved in the context of these two interaction components. These components use the display to channel their feedback. Distortion or disruption of their feedback, for example caused by an unreadable display, might result in more user actions towards the two components. Of course systems can also be designed based on other architectural ideas: about what defines a component, how they are connected, and how they communicate with the outside world. Still the description presented should give designers the outline to identify components in their system that could be evaluated by the proposed evaluation method.

3. Evaluation method

Like most empirical evaluation methods, activities of the proposed method can be categorised into four groups: preparation, execution, analysis and reporting. This section focuses on the first three activities by discussing the type of data that should be recorded, the test procedure, the measure, and the analysis of the data. At the core of the method lies a single component-specific measure that is based on the number of messages received by a component when participants accomplish a specific goal with the system. In the analysis phase this is set against optimal task execution. In contrast

to a holistic analysis, component-based analysis requires the evaluator to address the knock-on effect components might have on each other. Interaction problems with one component, for example understanding the red, yellow, green internet connect indicators (Fig. 2), could result in more interaction with another component, such as browsing through the radio list to look for alternative stations. Before describing the analysis in more detail by examining an example of a usability test session, the following section starts with setting out the method by describing the log file structure.

3.1 Log file recording

Table 1 gives an example of the content of a log file. Although evaluators might decide to record additional data, the Recipient and the Effort fields are essential for the analysis discussed later on. Starting with the ID field, this identification field uniquely identifies each recorded event. The Sender field is a reference to the component that sent the message, and the Recipient field refers to the component that received the message. A time stamp of when the event was received is recorded in the Time field. This time stamp could be in milliseconds, or seconds, depending on the precision desired. Note however that the reliability of the time stamp accuracy also depends on the type of operating system the application is running on (Myors, 1999). A description of the actual message is recorded in the Content field. Although outside the scope of the proposed method, this data could be relevant to conduct sequential data analysis such as lag sequential analysis, Fisher's cycles, and maximal repeated patterns (Sanderson and Fisher, 1997). Finally, the Effort field stores the physical interaction event effort associated with the creation of the messages. The analysis section (section 3.5) will look at the interpretation and the calculation of this value.

Table 1
Log file.

ID	Sender	Recipient	Time	Content	Effort
1	button	Station List	65100	Back	1
2	button	Station List	66100	Next	1
3	button	Station List	66600	Next	1
4	button	Station List	67804	Play	1
5	Station List	Receiver	67804	Play BBC 5	2

3.2 Test procedure

The test procedure of the testing method presented here fits well within the normal procedure of a usability test (e.g. Rubin, 1994), allowing therefore the method to run in parallel with the collection of other usability data such as thinking-aloud data, and task completion time. Participants are asked to complete a task while their interaction with the components is recorded in a log file. The task is finished once the participants attain a specific goal that would require them to alter the state of the interaction components under investigation. In advance, the participants should be instructed to act as quickly as possible to accomplish only the given goal, for example tuning into Metropol FM. Once they reach the goal, the recording stops, since new messages sent afterwards will probably be sent with a new goal in mind. When formulating a goal, engineers should also consider potential strategies participants may follow, including the generation of additional sub-goals when performing the task. For example, participants tuning into the Metropol FM station might decide also to add it to their

favourite station list. Although from an everyday perspective this seems a very efficient strategy, from an analysis perspective it complicates the interpretation of the additional effort. The engineer would have to determine whether this additional effort was caused by additional (sub-)goals that were intentionally set by participants, or caused by a less than efficient mapping of the main goal into sub-goals and resulting action selection and execution; a question, which seems impossible to answer without at least additional information. In the participants' briefing the engineer should, therefore, stress that they should only attempt to reach the goal set in the task description. Additionally asking future participants how and why they would use the device has been suggested (Cordes, 2001) to help the evaluator with formulating a task that matches real life goals. Of course evaluators also have to plan for the situation that a participant is unable to solve a task. They should set a threshold time, after which they intervene and help the participant. This threshold time could be based on data from a pilot study, for example the mean task completion time plus three times the standard deviation. This is a threshold that is often used in statistical analysis to find outliers.

3.3 Component-specific measure of the physical interaction event effort

ISO standard 9241-11 defines usability 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." (p. 3). The objective component-specific measure presented here only covers a small part of the broad usability spectrum by focussing on the efficiency dimension, "The resources expended in relation to the accuracy and completeness with which users achieve goals" (ISO, 1998, p. 3). Accuracy and completeness are regarded as constants because all participants are expected to achieve the same goal state in the usability test. Efficiency variation therefore only relates to resource expenditure. The component-specific measure is based on the concept of counting the number of messages an interaction component received directly, or indirectly via lower-level components. Only physical interaction events such as key presses are registered in the log file, hence the measure should only be associated with the physical interaction effort of creating these events. The scope of the efficiency dimension is narrowed down to *physical*, *interaction* and *event.* The effort does not involve the mental effort in sending a message or actions that are not communicated towards the system, for example changing between mouse and keyboard. Furthermore the measure is derived from discrete occurrences of receiving a message, and is therefore not a continuous variable such as task completion time. As mentioned before, each message represents a cycle of the control loop and therefore relates to the resource expenditure, or effort, to control the component. This idea has already been used to compare the efficiency of different versions of a component while the other components in the system remain the same, for example two similar calculators except for the size of the display screen (Brinkman et al., 2004a). However, the measure proposed here supports a single version and not a multiple versions testing paradigm. In the single version testing paradigm, only one version of each component is tested. The focus is to identify components that may hamper the overall efficiency of the system. Note that in the multiple versions testing paradigm, different versions of a component are compared with each other. The question in that case is which version has the highest efficiency, something that has been addressed elsewhere (Brinkman et al., 2005a). As the usability definition already indicates,

generalisation of usability findings depends on the generalisation of factors set in the evaluation, such as the user group, task, equipment, environment and the product. For the single version testing paradigm this also includes the generalisation of the components of a product. In other words, if a component performs well in a product it may not continue to do so after other components are changed in the product, or if the component is place in another product.

Comparing different components based on the number of messages received is only possible if two fundamental issues are addressed. First of all, a different amount of effort might be involved in sending a message to one component than to another component. For example, sending a message to the Station List component involves only a single button press, whereas sending a message to the Receiver component can involve a sequence of buttons to be pressed (e.g. next, next, next and play). The second issue is that efficiency problems that users encounter with one component could have a knock-on effect downwards on the number of messages received by a lower-level component. For instance, as mentioned earlier if users have problems understanding the feedback of the Receiver, it could result in more back, next and play messages sent to the Station List as they look for alternative radio stations.

3.4 Example of a usability test session

An example of a session from a usability test will help to explain how the method addresses these two issues. Imagine testing the web radio. Bob, the participant in the test, is given the task to tune into the Sports Week radio program, which is broadcasted on BBC 5. This last part of information however is kept from Bob. He is only told that the program is broadcasted on one of the BBC channels. As Bob tries to complete the task he encounters two classical problems. The first problem is about understandable progress information when users have to wait for the system to complete an action (Myers, 1985) and relates to the Receiver component. When Bob requests the BBC 5 station for the first time (Fig. 3, message 5), he fails to understand the meaning of the yellow indicator, signifying that the receiver is trying to establish a connection. Instead he concludes that BBC 5 is not available and moves on selecting BBC 6 (message 8). This time the Internet connection is established more quickly than before and the indicator also changes more quickly than before from yellow to green. After the streamed channel is played over the speaker, Bob wonders about the meaning of the red indicator, which until now has not lit up. He realises then that if this indicator lights up when the radio failed to establish a connection, he might have been too hasty with the BBC 5 station. He therefore decides to try the BBC 5 station again and waits a bit longer. As Fig 3 shows his second attempt is successful as afterwards the task is regarded as completed.

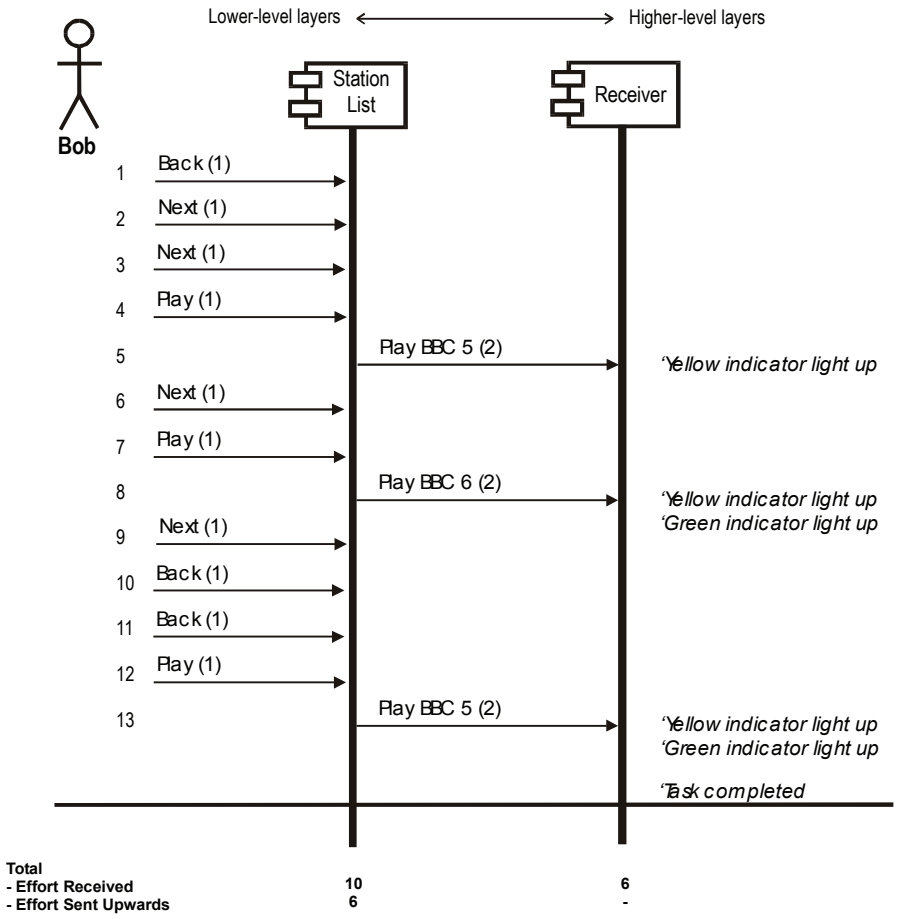

Fig. 3. Sequential diagram of the message exchange with the effort values given between brackets.

The second problem Bob encounters is a spatial compatibility problem (Sanders and McCormick, 1993) and relates to the design of the Station List component. From looking at the interface Bob has a problem with mapping the Back and Next buttons with moving the selection bar in the station list up or downwards. Fig. 3 shows that Bob has this problem on two occasions (see message 1 and 9). The first occasion is when he wants to move the bar downwards to select BBC 5 by pressing the Back button (message 1), which however moves the bar upwards. Bob undoes this by pressing the Next button (message 2). The second occasion where this problem recurs is when he wants to move back to BBC 5, but presses the Next button (message 9), and again recovers from this by pressing the Back button (message 10).

3.5 Analysing the physical interaction event effort

The first step in addressing the issue of the differences in the amount of physical interaction event effort involved when creating messages is to give every message a weight factor, referred to here in short as the effort value of a message. Fig. 3 shows that all the messages sent to the Station List component have been assigned a value one, representing one button press to create them. The messages sent from the Station List to the Receiver (message 5, 8 and 13) have an effort value of two. Each message could be created in principle by sending two messages to the Station List. Or more generally, the effort value assigned to a message (u_{n+1}) sent by a component (C) to a higher-level component, is the smallest possible summation of the effort value

assigned to messages needed to be received between this and the previous message (u_n) sent upwards. Notice the phrase "needed to be received". It is not the effort value of the messages actually sent by Bob, but that of messages that would be needed if the task were optimally executed, i.e. with the smallest amount of effort. Therefore, message 5 gets an effort value of two. Although Bob needed four messages (messages 1 to 4) to create this message, the same message could also have been created with only message 3 and 4. See the appendix for a more formal description of the assignment of weight factors.

With effort values assigned to all the messages, it is now possible to calculate the total extra effort (e) assigned to a component on top of the minimal effort assigned to a component if the task would be executed in an optimal manner. The latter can be found in the effort assigned to the message sent upwards. Therefore e for lower-level components is defined as:

$$e = \sum_{p \in P} p.\text{effort value} - \sum_{q \in Q} q.\text{effort value}$$

whereby

P = the set of messages received

Q = the set of messages sent upwards

This means that the total extra effort value (e) for the Station List is 10 - 6 = 4 (Fig. 3). This formula however, can not be used for the Receiver since it is the highest-level component, sending no message upwards. Instead e for the highest-level component(s) is defined as:

$$e = \sum_{p \in P} p.\text{effort value} - \sum_{o \in O} o.\text{effort value}$$

whereby

P = the set of messages received

O = the set of messages received if the task is optimally executed

To move the Receiver into the required state, Bob only had to send a Play BBC 5 message, which has an effort value of two. Therefore the e value of the Receiver is 6 - 2 = 4. Still, the e value for the Station List also includes the extra effort involved for sending messages to the Receiver caused by the problem Bob had with understanding the progress indicator of the Receiver. Therefore, a correction is needed for this downwards knock-on effect. One that estimates the effort as if the interaction with higher-level components was optimally executed (*norm*). This corrected value, called extra user effort (f), is defined as:

$$f = overhead \times norm$$

whereby

$$overhead = \frac{e}{\sum_{q \in Q} q.\text{effort value}} \quad \text{and} \quad norm = \sum_{o \in O} o.\text{effort value}$$

Q = the set of messages sent upwards

O = the set of messages received if the task is optimally executed

In the case of the Station List *f* would be 4 / 6 × 2 = 1⅓. The value 4/6 represents an *overhead* factor. In other words, the extra effort Bob puts into the interaction with the Station List is a factor ⅔ of what would be required. To put this factor into the context of a task, it is multiplied with 2, the effort value assigned to the component if the task was optimally executed. Therefore an *f* value of 1⅓ shows that Bob would have made approximately 1⅓ button presses extra to control the Station List than would have been theoretically required if his interaction with all other components had been optimally executed. For the Receiver, a top-level component, no correction is needed, therefore *f* = *e,* which means 4.

With the *f* values calculated, engineers can prioritise components according to the extra user effort Bob needed to interact with them. The Receiver component, with 4 extra keystrokes, seems a more problematic component to interact with for Bob than the Station List. For Bob, redesigning the Receiver might have a larger impact on the overall task execution than redesigning the Station List component. Although this example is relatively small with rather obvious problems, it shows how lower-level events, such as button presses, can be allocated to the interaction with higher-level components. The overall physical interaction event effort that went into the interaction with the web radio, as a single entity, is now redistributed to the components of the web radio. This also defines the scope of what the *f* values represent. They represent whatever the events at the lower level represent. In the example these were button presses, however other units are also possible as will be discussed in the next section.

3.6 Setting up and conducting the analysis

Before the actual analysis can take place, engineers have to 1) insert recording instructions into the software, 2) create a mechanism to calculate the message's effort value, and afterwards 3) collect the log file and calculate the *f* values for each component. The exact location to insert recording instructions depends on the program's architecture, and also on the programming style of the individual programmer. Just before a function or method call to another component seems the most obvious location. It is there in the execution that a message is sent to another component. For the lowest level components engineers should also insert recording instructions when components receive a message. A potential location to insert this code is the first lines of the function or a method.

As Table 1 shows, the recoded log of a message also includes the effort value associated with the optimal creation of the message. To determine the effort values engineers first have to establish the effort values when the task is executed in an optimal manner. As in the example, they need to start with assigning effort values to messages received at the lowest level. In the example that was a value of one for each button press. However, this might not always be appropriate or even possible, e.g. when the input is speech or gestures. In these cases it may be necessary to use a different unit to express the effort values of messages received in the lowest-level layer. They could for example be based on the minimal amount of time that would be needed to create particular lower-level messages, e.g. the number of seconds to utter a command. In other cases, it might be possible to rely on execution times of physical-motor operators such as pressing a button, or pointing a mouse, that have been defined for the Keystroke-Level model variant of the GOMS family (Card et al., 1983). Although the example assigned the same weight factor to a lowest-level message throughout the interaction, it is not required. Evaluators might want to increase the

accuracy of the measure by assigning different weight factors to similar messages to express factors such as fatigue or the starting position of the mouse or hand.

Once engineers have set the effort value of the message received by the lowest-level components, they have two ways to establish the effort values for the other messages. The first way is to write out the interaction as was done in the example. In other words, simulate the message exchange on paper. If the task is complex, involving the interaction between many components, this process can be relatively time consuming, comparable with conducting an extensive paper analysis of a GOMS Keystroke-Level model analysis (Card et al., 1980). The development of special computer-assisted tools, as exist for GOMS (Baumeister et al., 2000), could automate part of this analysis. The second way of establishing the effort values is quicker; however, it requires that recording instructions and some extra coding to calculate the effort values are already implemented. The idea is that the engineers simply execute the task with the device in the optimal manner and use the log file to obtain the effort values. To calculate the effort value the program has to add up the effort values of the message received by the component starting from the moment the previous message was set upwards.

Once the optimal effort values have been established for each message, engineers can include these values in the recording instructions. A complicating factor, however, is that effort values are not always static, but depend on the state a component was left in when the previous message was sent. For example, imagine that Bob started with his task with the selection bar of the Station List on ARL FM. The first *play BBC 5* message would have an effort value of 3, whereas the same message later on in the example would still have an effort value 2. In these cases engineers have to implement an algorithm that dynamically calculates the effort values based on the status of the component.

After the user trails have been completed, engineers need to collect the log files, and calculate *e*, *overhead*, *norm* and *f* values for each component. Standard computer tools such as a spreadsheet could help engineers to quickly calculate these values from potentially large amounts of log file data. As a final step, engineers could conduct statistical tests, such as *t*-test or *F*-test, to examine whether the *f* values of the components differ significantly.

Setting up and conducting this analysis involves time and effort from the engineers. This can be justified by the new insight they will gain from these *f* values. The values show the amount of effort that could be reduced if users interacted with the component in an optimal manner. Although a few extra button presses might not seem a great deal, Project Ernestine (Gray et al., 1993) has made people aware of the impact extra keystrokes can have if they are part of the critical path of processing, for example, a telephone call by a human operator. Gray et al. estimated that an average decrease of 1 second worktime per call would save their telephone company around $3 million a year. User effort has also been related to product sales numbers. For example Hahn's study (2001) of an online auction site showed that products attracted fewer visitors and consequently gained fewer bids if customers had to make more browse/search behaviours to locate the product's information. Engineers therefore can use the *f* values as indicators where potential improvements might be possible.

4. Validation

A vital step in developing a new efficiency evaluation method is its validation. In this case, does the suggested measure actually measure what it is supposed to measure, i.e. the physical interaction event effort to operate a component. Potential problems in validation studies of usability evaluation methods have created serious concerns in the human-computer interaction community about the effectiveness and validity of the results of these methods (Gray and Salzman, 1998). Therefore, this section provides a systematic attempt to assess the validity of the method. Four types of validity are often mentioned when it comes to empirical methods for social sciences (e.g. Neuman, 1997), these are: face validity, whether the measure looks valid; content validity, whether the full content of usability is represented in the measure; criterion validity, whether the results of the measure agree with other known measures; and construct validity, whether the *f* value measures the unobservable, theoretical construct —in full, the extra physical interaction event effort in controlling a specific component.

4.1. Face and content validity

There are no statistical procedures to analyse face or content validity. They are assessed by examining the measuring procedure and theoretical underpinning of the measure. In this case, the procedure seems acceptable, because similar procedures are used in usability tests, and the compositional approach taken is supported by previous research (e.g. Farrell et al., 1999; Haakma, 1999; Taylor, 1988). With regard to content validity, the scope of the measure has been defined acceptably narrow, involving physical interaction event effort, and making no claims about effort involved in mental activities or non-interaction related activities. Furthermore, the measure is an efficiency measure, and again makes no claims about other usability dimensions, such as satisfaction or effectiveness.

4.2. Criterion validity

Criterion validity can be split into two types: predictive validity, the ability to predict something it should theoretically be able to predict; and concurrent validity, the ability to give similar results as other accepted standard measures collected at the same time. Both can be assessed by applying statistical procedures on data gathered in an experiment. Therefore, an experiment with four prototypes of a mobile phone was set up. Besides collecting data to calculate *f* values, other usability data sets, such as task time, subjective usability, and interview transcripts, were also collected to assess concurrent validity. The experiment consisted of four usability tests, each examining the usability of one of the prototypes. The four prototypes were identical, except for the implementation of two components. To assess predictive validity, two different versions for each of these two components were used: an efficient and an inefficient version. This set-up gives an insight into *f* values as a measure to predict efficiency problems with an interaction component. Fig. 4 shows the architecture of the mobile phone. The two interaction components that were manipulated are printed in bold type. These components were responsible for the way participants could activate functions in the telephone (Function Selector, FS), and send text messages (Send Text Message, STM). Although the STM component was responsible for sending a text message, the experimental manipulation only related to the embedded sub-component called STM Main (STMM). This component was responsible for managing only the

dialogue for sending a text message and not for establishing the text or the telephone number, which was done by two other, embedded sub-components.

The efficiency variation between the versions of the FS and the STMM components related to the complexity of the dialogue structure that can be understood in terms of the Cognitive Complexity Theory (CCT) (Kieras and Polson, 1985). This theory holds that cognitive complexity increases when users have to learn more rules. To learn them, participants will have to engage in physical interaction to explore the behaviour of a component. For example, the efficient version of the STMM component guided the participants through the required steps of sending a text message. The inefficient version left the sequence of steps up to the participants. All of these options were presented as icons that forced the participants to learn the icon-option mapping rules. Furthermore, they also had to learn in which order to choose the options to execute that task in the right sequence.

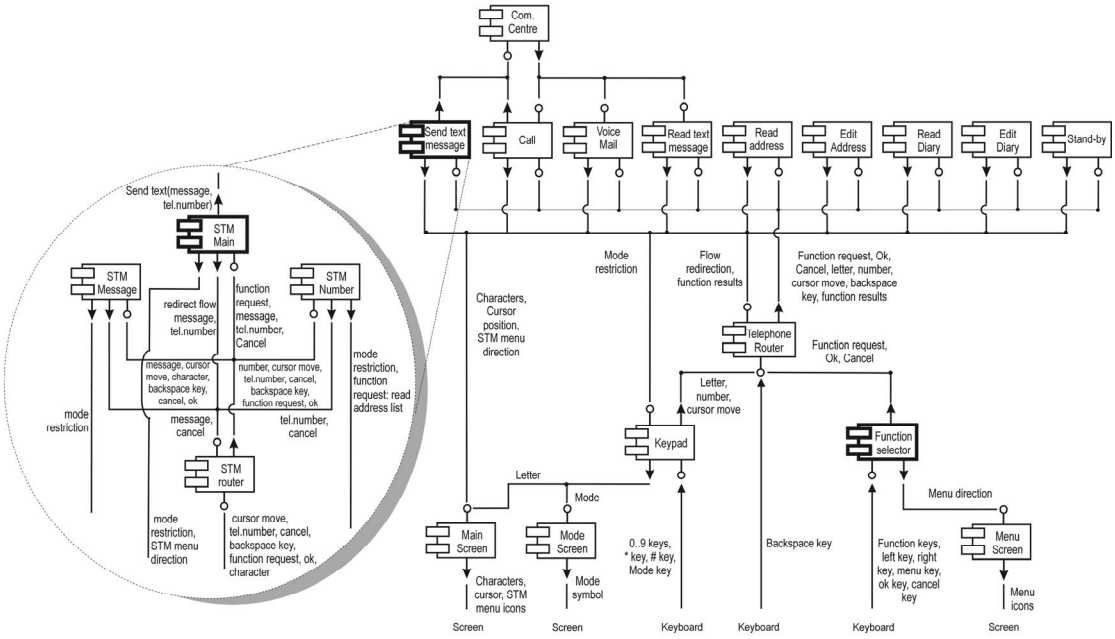

Fig. 4. The architecture of the mobile telephone with in bold the interaction components that were manipulated.

The efficient version of the Function Selector (FS) gave participants access to the telephone functions in a relatively broad but shallow menu, i.e. all eight options available within one stratum. In the inefficient 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 (Snowberry et al., 1983). In terms of CCT, the deep menu structure requires the participants to learn more rules to make the correct choices when going through the menu. Combining the versions of FS and STMM component led to four different mobile telephone prototypes.

The experimental environment was programmed in Delphi 5, and included PC emulators of all mobile telephones, a recording mechanism to capture the message exchange between the interaction components, and an automatic procedure to administer a usability questionnaire afterwards. All 40 participants were students of the Technische Universiteit Eindhoven. None of them had used a mobile telephone on

a daily or weekly basis[1]. The kinds of tasks they were required to perform with the mobile telephone were calling someone's voice-mail; adding a person's name and number to the phone's address list; and sending a text message. The application automatically assigned 10 participants to every prototype in a random order. At the end of the experiment, participants were asked to evaluate the mobile telephone and the two components separately with a questionnaire on the computer. The perceived overall usability and component-specific measures were all based on the six ease-of-use questions of the Perceived Usefulness and Ease-of-Use questionnaire (Davis, 1989) and on two satisfaction questions of the Post-Study System Usability Questionnaire (Lewis, 1995). The computer presented all questions in a random order. After completing the questionnaire participants were interviewed about the usability of the prototype and received NLG 22.50 (roughly €10) for their participation.

4.2.1. Predictive validity

The values of the *e* value, the *overhead* and *f* value were calculated for the Function Selector (FS) and the Send Text Message Main (STMM) component per prototype (Table 2). If the testing method was useful, it should be able to indicate whether engineers should focus their attention on the FS or STMM component in a particular prototype. Table 2 shows that the *f* value was above the mean *f* value of all prototypes when the version of a component was predicted to be inefficient to use, and below it when a version was predicted to be efficient to use. In other words, there were eight cases where the measure agrees with the theoretical prediction and no cases with disagreement, which is a significant result ($p. = 0.028$, Fisher's exact test).

Table 2
Mean *e*, *overhead* factor and *f* for two components in the four prototypes.

Components	Prototype				Overall
	1	2	3	4	Mean
Function Selector	Efficient[a]	Inefficient[a]	Efficient[a]	Inefficient[a]	
mean *e*	0.000	76.100[b]	0.000	100.500[b]	45.150
mean *overhead*	0.000	0.516[b]	0.000	0.600[b]	0.279
mean *f*	0.000	54.196[b]	0.000	62.983[b]	29.295
Send Text Message Main	Efficient[a]	Efficient[a]	Inefficient[a]	Inefficient[a]	
mean *e*	0.500	13.700	41.800[b]	78.100[b]	33.525
mean *overhead*	0.003	0.080	0.243[b]	0.363[b]	0.172
mean *f*	0.423	13.391	38.681[b]	66.450[b]	29.737

[a]predicted; [b]above the overall mean value of the 4 prototypes in that row.

The *f* values for prototype 2 and 3 also show how engineers can set their priorities. In the case of prototype 2, they should focus first on the FS component, because its *f* value is significantly ($t(9) = 3.60$, $p. = 0.006$, paired sample *t*-test) larger than the *f* value of the STMM component. As expected, in the case of prototype 3, it is exactly the other way around. The *f* value of the STMM component is significantly ($t(9) = -3.57$, $p. = 0.006$, paired sample *t*-test) larger than that of the FS component.

4.2.2. Concurrent validity

[1] 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.

Another type of criterion validity is concurrent validity. This was examined by comparing the *f* values with other usability measures such as task duration, number of keystrokes, questionnaire data, and interview transcripts of the debriefing. Table 3 shows significant correlations between the *f* values and the overall measures. The table also shows significant correlations between the *f* values of FS component and the component specific measures. The first measure, ease-of-use of FS, was created by using the Perceived Usefulness and Ease-of-Use questionnaire (Davis, 1989). However, instead of referring to the system in these questions, the questions referred to the menu. The same was done with the component-specific satisfaction questions. Besides collecting these questions, participants were also asked to fill out Norman's (1991) questionnaire on Menu Selection to evaluate the menu of the mobile phone. Again the results show that this measure significantly correlated with *f* values of the FS component. Another alternative measure was derived from the interviews conducted with the participants after the test. The table shows a significant correlation between the binary variable representing whether or not a participant mentioned a problem with the menu and the *f* values of FS component. A similar pattern of correlations was found for the *f* value of the STMM component. Although significant the sizes of the correlations with other component-specific measures were smaller. And the binary variable, problem reported with the STMM component in the interview debriefing, did not reach a significant level.

Table 3
Pearson correlations between *f* values of two component and other usability measures.

Measure	*f* of FS		*f* of STMM	
Overall				
Task duration	0.53	**	0.57	**
Extra keystrokes	0.55	**	0.73	**
Ease of Use	-0.69	**	-0.38	*
Satisfaction	-0.64	**	-0.22	
Component specific				
Ease of Use FS	-0.72	**	-0.33	*
Satisfaction FS	-0.70	**	-0.24	
Norman's Menu Selection questionnaire	-0.59	**	-0.28	
Reported FS problems in debriefing	0.62	**	0.07	
Ease of Use STMM	-0.49	**	-0.38	*
Satisfaction STMM	-0.29		-0.34	*
Reported STMM problems in debriefing	-0.00		0.27	

*p. <.05. **p. < .01

Since the *f* values in this experiment represented keystrokes it was possible to examine how effectively the overall number of keystrokes was distributed over the components. A regression analysis was therefore conducted. The analysis took as dependent variable the extra number of keystrokes made by the participants, and as independent variables the *f* values of the FS and STMM component. The analysis resulted in a significant model ($F(2,37) = 27.97$, $p. < 0.001$) that was able to explain 60% of the variation ($R^2 = 0.6$) in the extra number of keystrokes. Both the *f* values were significant coefficients in the model (Table 4). The values of the coefficients were close to one. This means that a change of one unit in the *f* values relates to a change of one unit in the overall number of extra keystrokes. Therefore, the calculation of the *f* value seems to result in a valid distribution of the lower-level events over higher-level components. Combined with the other results presented in this section it seems that the *f* measure has an acceptable level of both predictive and concurrent validity and consequently criterion validity.

Table 4
Estimated coefficients of regression model predicting the extra keystrokes made based on *f* values of FS and STMM component.

Model	*B*	Std. Error	Beta	*t*	*p.*
Constant	387.45	19.78		19.59	<0.001
f of FS	1.25	0.48	0.30	2.62	0.013
f of STMM	1.29	0.24	0.60	5.29	<0.001

4.3. Construct validity

The next analysis focuses on construct validity, or more specifically on discriminant validity, and examines whether the *f* value represents only the extra user effort related to a specific component. In other words, is an *f* value exclusively related to one component and not to others, or might a knock-on effect have intertwined them for example? Part of the answer can be found in Table 3. The *f* values of FS correlated with the ease-of-use rating of the STMM component and the *f* values of the STMM correlated with the ease-of-use rating of the FS component. Still with the exception of the ease-of-use rating of the STMM component, these correlations are smaller than the correlations between the *f* values and the related component-specific measures.

Besides examining the correlations with subjective measures, discriminant validity can also be examined by studying whether the version of the STMM component had an effect on the *f* values of the FS component. Therefore, an ANOVA was conducted on the data of the four prototypes. The version of the FS and the STMM component were the independent variables and the *f* values of the FS component was the dependent variable. As expected, the result showed a significant main effect ($F(1,36) = 94.87$, $p. < 0.001$) for the version of the FS component on the *f* values of the FS component. However, the results did not reveal a significant main effect ($F(1,36) = 0.53$, $p. = 0.470$) for the version of the STMM component, or a two-way interaction effect ($F(1,36) = 0.53$, $p. = 0.470$) between the versions of the two components on the *f* values of the FS component.

A similar analysis was carried out on the *f* values of the STMM component. An ANOVA was conducted with the same independent variables as before, however this time the dependent variable was the *f* value of the STMM component. Again as expected, the result showed a significant main effect ($F(1,36) = 4.75$, $p. = 0.036$) for the version of the STMM component. However, as before, no significant main effect ($F(1,36) = 0.96$, $p. = 0.337$) was found for the version of the other component, the FS component, or a two-way interaction effect ($F(1,36) = 0.16$, $p. = 0.726$) between the versions of the two components. Therefore, these results suggest an acceptable level of discriminant validity. The *f* values were mainly related to the efficiency of its component and not of other components.

5. Discussion

This study represents a detailed and systematic effort to examine a compositional approach to efficiency evaluation. The findings obtained in the experiment suggest that physical interaction associated with physical lower-level interaction events can successfully be attributed to higher-level components. This creates an individual efficiency measure for each component in the system. The validity of this measure

seems quite acceptable, and supports an evaluation approach that focuses on parts of the system and not only on the system as a whole. Take for example the results of prototype 3 in Table 2. Holistic measures such as task completion time could only say that the participants needed a specific time frame to complete the tasks and possibly set this against the time an expert would need. The component-specific measure gives more in-depth information. It allows engineers to see that the participants made more physical effort when interacting with the Send Text Message Main component than with the Function Selector component. This gives improvement effort a clear direction.

Although the extra user effort measure does not directly measure the extra physical interaction event effort in controlling a component, the results from the experiment suggest that the set of rules, for calculating a value for each component, can come up with a valid approximation. As with all evaluation methods, the described method also has its limitations. Firstly, the analysis only addressed a knock-on effect between components that works its way downwards; however a knock-on effect upwards is also possible. For instance an assumption is made that a component only receives and sends messages that are intended by the user. This however is not always the case. For example, the mobile phone in the experiment could also have been implemented with a Modified-Model-Position method instead of the Repeat-Key method to create characters. On average, users tend to find the Repeat-Key method easier to use than the other method (Detweiler at al. 1990). It involves having the users 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 Modified-Model-Position method involved having users first press either "*" or "#" key, depending on whether the letter is on the left or right position on the button label and nothing when the letter is in the middle. This is followed by a press on the key containing the letter (e.g. "*" followed by "GHI" for "G"). When users for the first time use the Modified-Model-Position, they often create unintended characters as they try to discover these rules (Brinkman, 2003), such as the characters in the middle of the label. Although users do not intend to create these characters, they create them in an optimal manner, which results in an e value that is too low. Furthermore, it also affects the components that receive these unintended messages, for example a String component that receives the characters. The problem related to the component responsible for the creation of characters is now transported to the String component, and consequently its e value becomes too high as all these messages are not required if the task is executed in an optimal manner.

A possible indicator that could help detecting these problems has been suggested (Brinkman, 2003). The indicator, called the standardised reception coefficient (s), is a ratio between two other ratios: the ratio between the observed number of messages sent upwards and received, and the ratio between the numbers of messages received and sent upwards in the case of optimal task execution, or more formally:

$$s = \frac{|Q|}{|P|} \times \frac{|O|}{|R|}$$

whereby

Q = the set of messages sent upwards

P = the set of messages received

O = the set of messages received if the task is optimally executed

R = the set of messages sent upwards if the task is optimally executed

If $s > 1$ engineers should study the component message stream. The user sent messages upwards with receiving on average fewer messages than optimally required. In other words, it seems that on average the user is out performing users that optimally execute the task, which by definition is impossible. Suggestions have been made to calculate the extra user effort value with another set of rules that would be less affected by these problems (Brinkman, 2003). However, it is still a topic open for further research, as a final solution which is not affected by unintended messages has not yet been established.

Besides the problem of unintended messages, the method does not take into account the fact that the efficiency of components can also be influenced by other components or factors such as the environment or the user. For example, because of poor usability, such as inconsistency (Brinkman et al., 2004b) or an overly demanding memory load (Brinkman et al., 2004a), users can start misusing other parts of a device while keeping the number of messages sent to the component under investigation the same.

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 (McLeod et al., 2005) automatically inserts recording code into a Java application without any need for access to the source code.

6. Final remarks

The current study confirms the possibility of efficiency testing in a CBSE environment. The direct benefit of the presented empirical testing method is the ability to evaluate at least a part of the usability spectrum of an individual component in a single system, something overall measures (e.g. task duration, number of keystrokes) cannot do in a big bang testing strategy. Furthermore, the method makes it possible to compare components in a single system based on their efficiency, which allows developers to prioritise their development effort. However, further research is still needed, especially when it comes to the real applicability of this approach. This will only become apparent when designers and developers put it into practice. It will then become visible how much extra effort and money is involved and how it fits in with normal working routines.

For interactive systems that are not developed according to the CBSE approach, the component-specific efficiency testing can still be useful once the control loops are identified. This would mean reverse engineering of a suitable compositional structure of the system. This compositional structure can be used to simulate the message exchange between the components. Lower-level messages, such as keystrokes, should

be fed into a conversion process that simulates the message exchange, which is again recorded in a log file. An example of such an approach can be found in the work of Docampo Rama (2001).

The testing method also has potential for efficiency testing outside the laboratory. However, further research will be needed on how to cope with everyday-life situations in which users use a system for multiple reasons over an extensive period of time. Field-based studies on components open the door to long-term efficiency studies, as a component would be a more practical study object than an entire application since components have a longer life span, as they can exist in multiple systems.

Acknowledgements

We would like to thank Rob van Eijk for the development of the mobile phone prototypes. We are also grateful for the comments and advice given by Audrey Bink, Nayna Patel and the anonymous reviewers, which helped us to improve the paper.

Appendix

The assignment of weight factors can also be described more formally by regarding a system A as set of components (CC), whereby the behaviour of each component can be defined as a finite state machine (e.g. Epp, 2004), such as $C = (S, R, U, v, \omega)$, where S is the set of states for C; R is the input alphabet of messages that can be received by C, and R^+ is the set of sequence of these messages; U is the output alphabet of messages that can be sent upwards by C; $v: S \times R^+ \rightarrow S$ is the state transition function; and $\omega: S \times R^+ \rightarrow U$ is the sent message upward function. The state of each component is a function of the component and the system state SS (λ: CC $\times$ SS $\rightarrow S$). Furthermore, the set of possible message sequences that a component can receive between the events when it sent a messages u_n and u_{n+1} upwards is a function of the component, the state of A when u_n was sent, and the state of A when u_{n+1} was sent (δ: $CC \times SS \times SS \rightarrow \wp(R^+)$). With this, the effort value of a message sent upwards (u_{n+1}) by component c, when the system is in state q, and whereby the system was in state p when the previous message was send upwards (u_n) by component c, can now be defined as:

$$u_{n+1}.\text{effort value} = \sum_{l \in x} l.\text{effort value}$$

whereby $x \in \delta(c, p, q)$ is called the minimal effort sequence to send u_{n+1} if

a) $\omega(\lambda(c, p), x) = u_{n+1}$

b) $y \in \delta(c, p, q)$ with $\omega(\lambda(c, p), y) = u_{n+1} \Rightarrow \sum_{k \in y} k.\text{effort value} \geq \sum_{l \in x} l.\text{effort value}$

This specification is flexible. It does not assume that messages received from lower-level components will always have the same effort value. The introduction in the specification of the δ function makes this possible. The function focuses only on the sequence of messages received between two events, the moment that the message was sent upwards and the moment that the previous message was sent upwards. Furthermore, each event is linked with the state of the entire system, and consequently it is also linked with the state of the lower-level components between these two events.

References

Aykin, N., 1994. Software reuse: A case study on cost-benefits of adopting a common software development tool, in: Bias, R.G., Mayhew, D.J. (Eds.), Cost-justifying usability. Academic Press, London, pp. 177-202.

Baumeister, L.K., John, B.E., Byrne, M.D., 2000. A comparison of tools for building GOMS models. The Proceedings of CHI 2000, 502-509.

Blandford, A., Green, T.R.G., Connell, I., 2005. Formalising an understanding of user-system misfits. in: Bastide, R., Palanque, P., Roth, J. (Eds.) Engineering human computer interaction and interactive systems. Springer, LNCS 3425, pp. 253-270.

Brinkman, W.-P., 2003. Is usability compositional? Doctoral dissertation, Technische Universiteit Eindhoven.

Brinkman, W.-P., Haakma, R., Bouwhuis, D.G., 2004a. Memory load: A factor that links the usability of individual interaction components together. Proceedings of HCI 2004, vol. 2. Research Press international, Bristol, United Kingdom, pp.165-168.

Brinkman, W.-P., Haakma, R., Bouwhuis, D.G., 2004b. Consistency: A factor that links the usability of individual interaction components together. The Proceedings of ECCE-12. European Association of Cognitive Ergonomics, France, pp .57-64.

Brinkman, W.-P., Haakma, R., Bouwhuis, D.G., 2005a. Empirical usability testing in a component-based environment: Improving test efficiency with component-specific usability measures, in: Bastide, R., Palanque, P., Roth, J. (Eds.) Engineering human computer interaction and interactive systems. Springer, LNCS 3425, pp. 20-37.

Brinkman, W.-P., Haakma, R., Bouwhuis, D.G., 2005b. Usability Testing of Interaction Components: Taking the message exchange as a measure of usability, in: Jacob, R.J.K., Limbourg, Q., Vanderdonckt, J. (Eds.), Computer-Aided Design of User Interfaces IV. Kluwer Academic, Dordrecht, The Netherlands, pp. 159-170.

Broekman, B., Notenboom, E., 2003. Testing embedded software. Addison-Wesley, London.

Card, S.K., Moran, T.P., Newell, A., 1980. The keystroke-level model for user performance time with interactive systems. Communications of ACM 23 (7), 396-410.

Card, S.K., Moran, T.P., Newell, A., 1983. The psychology of human-computer interaction. Lawrence Erlbaum, London.

Carver, C.S., Scheier, M.F., 1998. On the self-regulation of behavior. Cambridge University Press, New York, NY.

Chin, J.P., Diehl, V.A., Norman, L.K., 1988. Development of an instrument measuring user satisfaction of the human-computer interface. Proceedings of CHI'88, ACM press, New York, NY, pp. 213-218.

Cordes, R.E., 2001. Task-selection bias: a case for user-defined tasks. International journal of human computer interaction, 13(4), 411-419.

Coutaz J., 1987. PAC, an object oriented model for dialog design. Proceedings of INTERACT'87. North-Holland, Amsterdam, pp. 431-436.

Cox, B.J., 1990. There is a silver bullet: A software industrial revolution based on reusable and interchangeable parts will alter the software universe. Byte 15 (10), 209-218.

Davis, F.D., 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Quarterly 13 (3), 319-340.

Davis, F.D., Venkatesh, V., 2000. A theoretical extension of the technology acceptance model: four longitudinal field studies. Management Science 46 (2), 184-204.

Detweiler, M.C., Schumacher, M.C., Gattuso, N.L., 1990. Alphabetic input on a telephone keypad. Proceedings of the Human Factors Society – 34th Annual Meeting, vol. 1. Human Factors Society, Santa Monica, CA, pp. 212-216.

Docampo Rama, M., 2001. Technology generations handling complex user interfaces. Doctoral dissertation, Technische Universiteit Eindhoven, The Netherlands.

Dragivevic, P., Fekete, J.-D., 2001. Input device selection and interaction configuration with icon. Proceedings of IHM-HCI. Springer Verlag, Lille, France, pp. 553-558.

Duke, D., Faconti, G., Harrison, M., Paternò, F., 1993. Unifying views of interactors. Proceedings of the Workshop on Advanced Visual Interfaces, ACM Press, pp. 143-152.

Epp, S.S., 2004. Discrete mathematics with applications (3rd ed.) Thomos-Brooks/Cole, Belmont, CA.

Farrell, P.S.E., Hollands, J.G., Taylor, M.M., Gamble, H.D., 1999. Perceptual control and layered protocols in interface design: I. Fundamental concepts. International Journal of Human-Computer Studies 50 (6), 489-520.

Gram, C., Cockton, G., 1996. Design principles for interactive software. Chapman & Hall, London.

Gray, W.D., John, B.E., Atwood, M.E., 1993. Project Ernestine: Validating a GOMS analysis for predicting and explaining real-world task performance. Human-computer interaction, 8, 237-309.

Gray, W.D., Salzman, M.C., 1998. Damaged merchandise? A review of experiments that compare usability evaluation methods. Human-computer interaction 13 (3), 203-261.

Haakma, R., 1998. Layered feedback in user-system interaction. Doctoral dissertation, Technische Universiteit Eindhoven, The Netherlands.

Haakma, R., 1999. Towards explaining the behaviour of novice users. International Journal of Human-Computer Studies 50 (6), 557-570.

Hahn, J., 2001. The dynamics of mass online marketplaces: A case study of online auction. Proceedings of CHI'01, ACM Press, New York, NY, pp. 317-324.

Hertzum, M., Jacobsen, N.E., 2001. The evaluator effect: A chilling fact about usability evaluation methods. International Journal of Human-Computer Interaction 13 (4), 421-443.

Hilbert, D.M., Redmiles, D.F., 2000. Extracting usability information from user interface events. ACM Computing Surveys 32 (4), 384-421.

Internal Organization for standardization (ISO), 1998, ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) -- Part 11: Guidance on usability. Geneva.

John, B.E., Bass, L., Sanchez-Sequra, M.-I., Adams, R.J., 2005. Bringing usability concerns to the design of software, in: Bastide, R., Palanque, P., Roth, J. (Eds.) Engineering human computer interaction and interactive systems. Springer, LNCS 3425, pp. 20-37.

John, B.E., Marks, S.J., 1997. Tracking the effectiveness of usability evaluation methods. Behaviour & Information technology 16 (4/5), 188-202.

Kieras D., Polson P.G., 1985. An approach to the formal analysis of user complexity. International Journal Man-Machine Studies 22 (4), 365-394.

Krasner, G.E., Pope, S.T., 1988. A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80. Journal of Object-Oriented Programming 1 (3), 27-49.

Lewis, J.R., 1995. IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. International Journal of Human-Computer Interaction 7 (1), 57-78.

McLeod, I., Evans, H., Gray, P., Mancy, R., 2004. Instrumenting bytecode for the production of usage data, in: Jacob, R.J.K., Limbourg, Q., Vanderdonckt, J. (Eds.), Computer-Aided Design of User Interfaces IV. Kluwer Academic, Dordrecht, The Netherlands, pp 185-195.

Molich, R., Ede, M.R., Kaasgaard, K., Baryukin, B., 2004. Comparative usability evaluation. Behaviour & Information Technology 23 (1), 65-74.

Myers, B., 1985. The importance of percent-done progress indicators for computer-human interfaces. Proceedings of CHI'85. ACM Press, New York, NY, pp. 11-17.

Myors, B., 1999. Timing accuracy of PC programs running under DOS and Windows. Behavior Research Methods, Instruments, & Computers 31(2), 322-328.

Neuman, W.L, 1997, Social research methods: Qualitative and quantitative approaches. Allyn and Bacon, Boston, MA.

Newell, A., 1990. Unified theories of cognition. Harvard University Press, London.

Nielsen, J., 1986. A virtual protocol model for computer-human interaction. International Journal of Man-Machine Studies 24 (3), 301-312.

Nielsen, J., Molich, R., 1990. Heuristic evaluation of user interfaces. Proceedings of CHI'90, ACM Press, New York, NY, pp.249-256.

Norman, D.A., 1984. Stages and levels in human-machine interaction. International Journal of Man-Machine Studies 21 (4), 365-375.

Norman, K.L., 1991. The psychology of menu selection: Designing cognitive control of the human/computer interface. Ablex Publishing corporation, Norwood, NJ.

Paternò, F., 2000. Model-based design and evaluation of interactive applications. Springer, London.

Polson, P.G., Lewis, C., Rieman, J., Wharton, C., 1992. Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. International journal of Man-Machine Studies 36 (5), 741-773.

Powers, W.T., 1973. Behavior: The control of perception. Aldine Publishing Company, Chicago, IL.

Rubin, J., 1994. Handbook of usability testing: how to plan, design, and conduct effective tests. Wiley, New York, NY.

Sanders, M.S., McCormick, E.J., 1993. Human factors in engineering and design. McGraw-Hill, New York, NY.

Sanderson, P.M., Fisher, C., 1997. Exploratory sequential data analysis: qualitative and quantitative handling of continuous observational data, in: Salvendy, G. (Ed.), Handbook of human factors and ergonomics, John Wiley & Sons, Inc., Chichester, pp. 1471-1513.

Taylor, M.M., 1988a. Layered protocol for computer-human dialogue. I: Principles. International Journal of Man-Machine Studies 28 (2-3), 175-218.

Taylor, M.M., 1988b. Layered protocols for computer-human dialogue. II: some practical issues. International journal of Man-Machine Studies 28 (2-3), 219-257.

Taylor, M.M., 1989. Response timing in layered protocols: A cybernetic view of natural dialogue, in: Taylor, M.M., Néel, F., Bouwhuis, D.G., (Eds.) The structure of multimodal dialogue II. John Benjamins, Amsterdam, pp. 159-172.

Taylor, M.M., Farrell, P.S.E., Hollands, J.G., 1999. Perceptual control and layered protocols in interface design: II. The general protocol grammar. International journal of Human-Computer Studies 50 (6), 521-555.

Taylor, M.M., Waugh, D.A., 2000. Multiplexing, diviplexing, and the control of mutimodal dialogue, in: Taylor, M.M., Néel, F., Bouwhuis, D.G., (Eds.) The structure of multimodal dialogue II. John Benjamins, Amsterdam, pp. 439-456.

Vallacher, R.R., Wegner, D.M., 1987. What do people think they're doing? Action identification and human behavior. Psychological Review 94 (1), 3-15.