Human-Computer Interaction —INTERACT'01
Michitake Hirose (Ed.)
Published by IOS Press, © IFIP, 2001

Usability evaluation of component-based user interfaces

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

IPO, Center for User-System Interaction, Technische Universiteit Eindhoven,
PO Box 513, 5600 MB Eindhoven, the Netherlands

w.p.brinkman@tue.nl, r.haakma@tue.nl, d.g.bouwhuis@tue.nl

Abstract: The idea of building a device out of separate parts is gaining more common practice in the software industry. Software parts are designed with the potential for reuse. The reuse of the software parts promises to reduce development cost and time. This process triggers some new usability issues, especially when the user interface (UI) is also designed as a complex of several parts. A study is carried out to find out whether and how the usability of a UI part can be tested, and thirdly how the usability of an individual UI part can be affected by other UI parts. Analysis of log files that captured the basic user-system interaction components seems a promising evaluation method. Preliminary experimental results show that this evaluation method is more sensitive than traditional method as number of keystrokes or task time analysis.

Keywords: interaction component, usability evaluation, log file analysis

1 Introduction

An increasing number of software applications have a component-based design. With components, software engineers can encapsulate pieces of a system, which can be developed individually. Software engineers who make use of these components only need to know the external interface of the component to be able to use its functionality. The component's internal details are hidden. This makes it possible for components to be replaced by other components as long as they provide the same external interface. Furthermore, components can be reused in different applications. This creates the possibility of reducing the development time and thereby costs. These benefits are nevertheless based on a very much system-centred view, which may degrade the system usability (Hertzum, 2000), since a lot of system functionality may be provided by components developed with other situations of use in mind.

The user interface software itself can also be constructed of components. An elementary component of a mobile phone may receive the keystrokes and pass them on as characters to another component. This component may collect the characters into a string and allow additional editing. When the user finishes entering the string, the component sends the complete string to another component that adds a telephone number to compose a text message that can be sent. This user-system interaction is described by the Layered Protocol Theory (Farrell et al, 1999). It portrays the user-system interaction on different levels. User–system interaction on the highest level is established by mediation of user-system interaction on lower levels. The interaction on each level may follow a different protocol.

Instead of the complete user interface, Haakma (1998) suggested that the usability of individual components could be studied. Components that hamper the overall interaction could be pinpointed and replaced by more useable ones. Ready-made usable components could be made for a specified user group to achieve a specified goal within in a specified context of use. Software engineers could use them to create new usable user interfaces. This raises the following questions, namely: whether and how the usability of components can be tested, and thirdly how the usability of an individual component can be affected by other components. Only the first two questions will be addressed here.

2 Usability testing

The user-system interaction can be seen as an exchange of messages between the user and components. Although the user can not directly send and receive messages to and from the high level components, the messages passed on by lower level components to higher level components and vice versa can be regarded as virtual messages between the user and the high level components. While recording these messages in a log file facilitates study of the user-component interaction, a precise definition is needed of a component that can be analysed with a log file. Vital for the usability evaluation is that users somehow should be able to determine the state of a component and be able to change it too. Without the possibility of perception, users can not separate the component from the whole system. Without the ability to control their perception, their behaviour is pointless. Therefore, the term *interaction component* is introduced. An interaction component is defined as a unit within a device that receives direct or indirect signals from the user, which enable the user to change the state of the interaction component. The user is by definition able to perceive or to infer the state of the interaction component.

To compare the efficiency of different versions of an interaction component the following experimental paradigm can be used. Users are observed while they perform a task to achieve a specific goal with different versions of a system. Applying different versions of the interaction component in the system and leaving the rest of the system the same creates these system versions. The users have to reach the goal, and thereby at least alter the state of the interaction component. They are instructed to do so as fast as possible. All messages sent to the interaction component are recorded in a log file.

Based on a first explorative study the hypothesis is put forward that within this paradigm the interaction component version that received the fewest messages is the most efficient one. The number of messages indicates the users' effort to control their perception of the interaction component. It is assumed that the users' effort to send a single message is equal in each version.

A second experiment was carried out to validate this hypothesis. Eight prototypes of a mobile phone were created by applying one out of two versions of the following three interaction components: the Function Selector, the Keypad, and the Send Text Message Function. The two versions were designed to differ with respect to efficiency. For example in the case of the Send Text Message Function, the presumably more efficient version allowed the user to directly enter a text message and afterwards automatically requested the telephone number. Whereas in the less efficient version the users were first confronted with two options, to send the message or to edit it. The second option would lead users to two new options: edit the text message or the telephone number. When the users, for instance, choose to edit the text message they could enter the text. Afterwards the users were brought back to the first two options, requiring them again to choose for the edit option and consequently in this case for the phone number option. Only when both text message and telephone number were given, the users could send the message with the send option. All these options were presented as icons that forced them to learn the icon-option mapping.

independent variable	correctly classified	Wilks' lambda	df	sig.
messages	88%	0.499	1	0.000
task time	58%	0.972	1	0.137
keystrokes	55%	0.992	1	0.430

Table 1: Results of the discriminant analyses with the version of the Send Text Message Function as dependent variable.

The preliminary results confirm the hypothesis and even indicated that the number of messages received by interaction components is a more sensitive measure than the traditional number of keystrokes or task time analysis. For example, Table 1 shows that only with the number of messages received by Send Text Message Function interaction component as independent variable significantly more of the eighty users were linked with the employed interaction component's version than a random classification of 50%.

Future research will look more at the third question, how the usability of an individual component can be affected by other components.

References

Haakma, R., 1998. *Layered Feedback in User-System Interaction.* Thesis (PhD). Technische Universiteit Eindhoven.

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.

Hertzum, M. 2000. Component-Based Design May Degrade System Usability: Consequences of Software Reuse. *In:* C. Paris, N. Ozkan, S. Howard, S. Lu eds. *OZCHI 2000 Conference Proceedings*, Sydney (Australia): CHISIG, 88-94.