

Is usability compositional?

Citation for published version (APA):

Brinkman, W. P. (2003). *Is usability compositional?* [Phd Thesis 1 (Research TU/e / Graduation TU/e), Industrial Engineering and Innovation Sciences]. Technische Universiteit Eindhoven. <https://doi.org/10.6100/IR562468>

DOI:

[10.6100/IR562468](https://doi.org/10.6100/IR562468)

Document status and date:

Published: 01/01/2003

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Is usability compositional?

Willem-Paul Brinkman

The work described in this thesis has been carried out under the auspices of the J.F. Schouten School for User-System Interaction Research.

© 2003 Willem-Paul Brinkman - Eindhoven - The Netherlands.

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Brinkman, W.-P.

Is usability compositional? / by Willem-Paul Brinkman. — Eindhoven: Technische Universiteit Eindhoven, 2003. — Proefschrift. —

ISBN 90-386-1947-2

NUR 811

Keywords: Human-computer interaction / Usability / Usability evaluation methods / Sequential data analysis/ Component-based software engineering

Printing: Universiteitsdrukkerij Technische Universiteit Eindhoven.

Is usability compositional?

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr. R.A. van Santen, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op woensdag 15 januari 2003 om 16.00 uur

door

Willem-Paul Brinkman

geboren te Middelburg

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr. D.G. Bouwhuis

en

prof.dr. P.M.E. De Bra

Copromotor:

dr.ir. R. Haakma

Acknowledgements

The research described in this thesis was conducted at Eindhoven University of Technology, first at IPO Centre for User-System Interaction, and later on at the Department of Technology Management. During those four and a half years, many people contributed to this work. First of all, I thank Don Bouwhuis and Reinder Haakma for their advice and discussions, which were always stimulating and helped me to clarify many issues.

Working with Maxim Schram on the modelling of a TV set was very productive. Having a more or less real product to describe helped me to understand the complexity of it. Thanks for going through this ordeal with me. I am also indebted to Rob van Eijk, who programmed and pre-tested the mobile telephone simulation described in chapter 4, and Willem Peeters, who set up, conducted and analysed an earlier version of the experiment described in chapter 6. Mark Houben's involvement provided the different simulations with the right sounds, but also his Matlab expertise is highly appreciated. Special thanks go to Arnout Fischer and Audrey Bink for carefully reading drafts of this thesis. Without them I would certainly consider reading this thesis a challenge.

Furthermore, I would like to express my gratitude to Michel Alders, Agnieszka Matysiak, and Oleksandr Krashtan for being involved in their final projects of the User System Interaction Programme. Their work brought me into closer contact with the problems faced by designers and software engineers when it comes to creating a user interface.

Finally, I want to thank the IPO and MTI members. They were without a doubt parties to the good atmosphere. It was a delight doing this Ph.D. research. We should do it again some time.

Willem-Paul Brinkman

Contents

1	Introduction	1
1.1	Devices and usability	1
1.1.1	The user interface: one or many?	2
1.2	Layered Protocol Theory	2
1.3	Component-based software engineering	5
1.3.1	Component-based software development	6
1.3.2	Software architecture	6
1.4	Usability evaluation	9
1.5	The research objectives	11
1.5.1	An empirical component-based testing framework with a theoretical background	11
1.5.2	Is the whole more than the sum of its parts?	12
1.6	Overview thesis	12
2	An explorative study on control and the comprehension of user interface components	15
2.1	Introduction	15
2.1.1	Description message exchange between components	15
2.1.2	Objectives	17
2.2	Method	18
2.2.1	Prototype	18
2.2.2	Experimental design	21
2.2.3	Training	22
2.2.4	Hypotheses	25
2.2.5	Procedure	28

2.3	Results	28
2.3.1	Control tests	28
2.3.2	Overall performance	30
2.3.3	Control on the lower-level layer	31
2.3.4	Control on the higher-level layer	34
2.4	Discussion	35
2.5	Conclusions	37
3	Usability testing in a component-based environment	39
3.1	Introduction	39
3.1.1	Objectives evaluation method	39
3.1.2	Overview of the evaluation method	40
3.1.3	Overview chapter	40
3.2	Formalising LPT	41
3.2.1	Architecture	41
3.2.2	Control effects	44
3.3	Evaluation method	46
3.3.1	Test procedure	46
3.3.2	Objective performance measure in the multiple versions testing paradigm	47
3.3.3	Objective performance measure in the single version testing paradigm	49
3.3.4	Subjective measures	59
3.4	Comparison with other evaluation methods	60
3.4.1	Sequential data analysis	60
3.4.2	Not log-file-based usability evaluations	61
4	An experimental evaluation of the component-based usability testing framework	63
4.1	Introduction	63
4.2	Method	64
4.2.1	Prototypes and operationalisation claims	65
4.2.2	Design	74
4.2.3	Procedure	75
4.3	Results	75

4.3.1	Data preparation	76
4.3.2	Multiple versions testing paradigm	77
4.3.3	Single version testing paradigm	86
4.4	Discussion	90
4.4.1	The power of the objective component-specific performance measure	92
4.4.2	The power of subjective component-specific usability measures . . .	92
4.4.3	Objective component-specific performance measure in the single ver- sion testing paradigm	93
4.4.4	Ineffectiveness problems	93
4.4.5	General remarks	94
4.5	Conclusions	95
5	Effects of consistency on the usability of user interface components	97
5.1	Introduction	97
5.1.1	Consistency and LPT	97
5.2	General experimental set-up	99
5.3	Experiment 1 —consistency within the same layer	100
5.3.1	Method	100
5.3.2	Results	102
5.3.3	Discussion	109
5.4	Experiment 2 —consistency between lower-level and higher-level layers . .	110
5.4.1	Method	111
5.4.2	Results	114
5.4.3	Discussion	122
5.5	Experiment 3 —consistency between the component and the application domain	124
5.5.1	Method	124
5.5.2	Results	128
5.5.3	Discussion	132
5.6	General discussion	134
5.6.1	Theoretical implications	134
5.6.2	Practical implications	135
5.7	Conclusions and further research	136

6	Effect of mental effort on the usability of user interface components	137
6.1	Introduction	137
6.2	Method	138
6.2.1	Prototypes	138
6.2.2	Hypothesis	139
6.2.3	Alternative hypothesis	140
6.2.4	Tasks	141
6.2.5	Measures	142
6.2.6	Experimental design	145
6.2.7	Subjects	145
6.2.8	Procedure	145
6.3	Results	146
6.3.1	Data preparation	146
6.3.2	Statistical analyses	148
6.4	Discussion and conclusions	152
6.4.1	Interpretation results	152
6.4.2	Theoretical implications	155
6.4.3	Practical implications	157
7	Discussion and conclusions	159
7.1	Recapitulation	159
7.2	Main conclusions and limitations	160
7.2.1	Component-based usability testing	160
7.2.2	The independence of the component's usability	163
7.3	Implications	164
7.3.1	Theoretical implications	164
7.3.2	Practical implications	166
7.4	Future work	167
	Bibliography	169
	Appendix	179

A Formal specification objective performance measure in the single version testing paradigm	181
B Standard questions used in the questionnaires	185
Summary	187
Samenvatting (Summary in Dutch)	191
Curriculum vitae	195

Chapter 1

Introduction

1.1 Devices and usability

In the past few decades, people have surrounded themselves with a growing number of devices that offer increasingly more complex functionality. Consequently, the devices' usability has had a growing effect on areas like the military, economy, society and politics. The Second World War led to the birth of human factors, or ergonomics, as a research field. From then on, aircraft cockpits, sonar, radar, etc. were increasingly designed in accordance with the capabilities of soldiers. The army and navy could no longer solely depend on selection and training to cope with the peculiarities of devices. In such a period of national emergency, selection standards were relaxed and equipment was operated by half-trained soldiers (Driskell & Olmstead, 1989; Meister, 1999). In the civilian society, product developers recognised that they could compete with the usability of products. So they set out to develop devices consumers could easily use. Sales figures dropped when consumers had to struggle with the supporting artefacts of a service as online shopping (Lohse & Spiller, 1998) or online auctions (Hahn, 2001). Concerns have also been expressed about a possible division in society between those who can handle complex artefacts and those who can not. Some press for universal usability because they believe that in a fair society all individuals would have an equal opportunity to participate in, or benefit from, the use of computer resources regardless of race, sex, religion, age, disability, national origin or other factors (Shneiderman, 2000). This is becoming increasingly important, as governments rely more and more on electronic means to communicate with their citizens. Even democratic processes are disrupted by bad usability since part of the controversy surrounding the 2000 United States presidential election concerned the confusing butterfly ballot format (Sinclair, Mark, Moore, Lavis, & Soldat, 2000). In Palm Beach County, Florida, people who had intended to vote for Al Gore might have end up voting for Pat Buchanan by mistake. All these examples illustrate that theories that explain and predict the interaction between humans and systems are very welcome indeed, because they can guide designers in their aspiration to design more easy-to-use artefacts.

1.1.1 The user interface: one or many?

This thesis deals with the question whether usability is compositional. Artefacts can be approached in two ways, holistically, or as a collection of components. Elementary components such as pop-up menus, radio buttons, list boxes, give access to more abstract components, or are embedded in other more complex components, such as spelling checkers or email components in word processors or spreadsheets. Developers hope that applying highly usable ready-made components will result in a highly usable device. Still, when it comes to studying the usability of a device, so far, the usability of those components has not been assessed individually, but only for their impact on the overall usability. Therefore, the focus here is on whether and how the usability of components can be tested, and how the usability of an individual component can be affected by other components. One theory that uses such a division of an artefact is the Layered Protocol Theory (LPT), which provides the basis for usability testing in a component-based software engineering approach. In the following sections this idea will be further explained.

1.2 Layered Protocol Theory

LPT decomposes the user-system interaction into different layers that can be designed and analysed separately. The theory was put forward by Taylor in 1988, but the basic ideas were already mentioned earlier in a report on spatial information systems (Taylor, McCann, & Tuori, 1984). The theory was inspired by the ISO (1994) Open Systems Interconnect reference model for communication among computers, but also has its origins as early as in the work of Donders in 1862 and Wundt in 1880 (Taylor, 1988a). Later on, LPT was recognised (Farrell, Hollands, Taylor, & Gamble, 1999) as a special form of Powers' (1973, 1998) Perception Control Theory (PCT). Whereas PCT is a general theory about human interaction with the environment, LPT focuses on the human-human and human-system communication and interaction. Central concepts in these theories are the perception-control loop and the accumulation of these control loops, which creates multiple layers in which the interaction takes place.

The main tenet of PCT is that the purpose of any given behaviour is to control perception. This differs from both traditional behaviouristic and cognitive theories (Bourbon & Powers, 1999; Chéry & Farrell, 1998; Farrell et al., 1999). Behaviour in traditional behaviouristic Stimulus-Response theories had no purpose and was controlled by the environment. It was simply produced since it was linked to a particular stimulus. There was no admission that something inside a person might be responsible for behaviour, which specifically is the case according to the cognitive view. In the traditional cognitive view, behaviour was the result of an internal planning process. However, the result of the planning process could be out of date at the moment of execution as the world could change during the planning phase. In PCT behaviour is not planned but continuously adjusted to control the perception of the changing world according to an internal reference value. Today, however,

these views of behaviouristic and cognitive theories are outdated. The idea of purpose has been the hallmark of modern behaviorists (Herrnstein, 1967) introducing action selection by consequences. On the other hand, cognitive theories, such as Activity theory, Situated Action Models, and Distributed Cognition, no longer separate individuals from the work context, as the theories look at the setting, the social group and the supporting artefacts (Nardi, 1996).

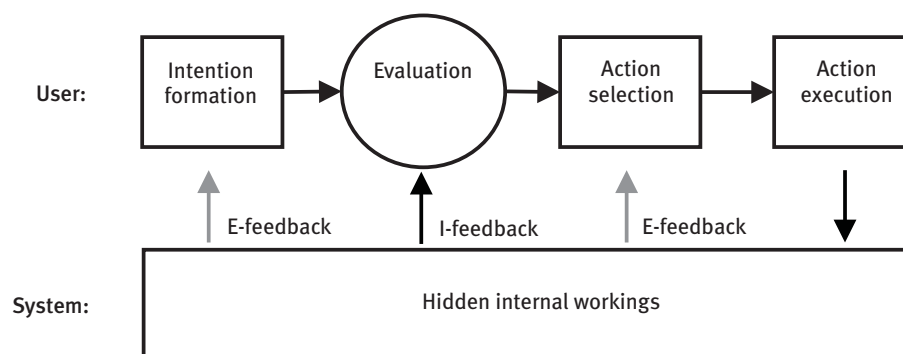


Figure 1.1: Perceptual-control loop. The four stages are shown from the perspective of the user’s side of the loop. The system is shown as a kind of black box hiding its internal workings. Users can only observe the Expectation and Interpretation feedback provided by the system. By executing actions, like pressing buttons, users can make the system change its interpretation feedback (Haakma, 1999).

Figure 1.1 presents the perceptual-control loop between a user and a system according to Haakma (1999). On the user side of the control loop, the following four stages (Norman, 1984) can be identified: intention formation, evaluation, action selection, and action execution. First, users are expected to develop some ideas about what they want to perceive. A combination of higher-level reference values, knowledge, and system feedback on what is achievable with the system (expectation feedback: E-feedback) guides users to establish the appropriated intentions. For instance, watching a TV set may cause users want to see the weather forecast to learn if they will stay dry when they go outside. In the next stage, the users evaluate whether the system meets their intentions; does the TV set show the weather forecast? The feedback about the systems’ current state, reflecting how the system interpreted previous user actions, is called interpretation feedback (I-feedback). When the users’ perception deviates from their intentions, they can decide to act upon the system to overcome this, which brings them to the next stage. Here, the users select physical actions or create new sub-intentions (switching the TV set on, and selecting the right channel) to remove the discrepancy. The E-feedback again guides users at this stage; the buttons’ labels on the TV provide the set of actions to choose from. In the final stage, users execute the selected actions. This can mean going through a loop at a lower-level layer, or actually performing physical actions, which make up the system’s input.

Within LPT, user actions and system feedback are regarded as an exchange of messages.

The set of communication rules the user and system have to apply to get the messages across, is called a protocol. Work has been done to specify a general protocol grammar that describes the way in which possible communication errors are avoided or corrected (Taylor, 1988b; Taylor, Farrell, & Hollands, 1999). In addition to grammatical problems, the time delay between sending a user messages and receiving I-feedback could also lead to communication problems (Taylor, 1988b, 1989). If the TV set does not provide a perceivable response fast enough when it is switched on, users may repeat the action, which would in fact switch the TV set off again. Summarising, in LPT, users send messages to control the I-feedback and both the I- and E-feedback present the system's contribution to strengthening this control (Engel, Goossens, & Haakma, 1994; Engel & Haakma, 1993).

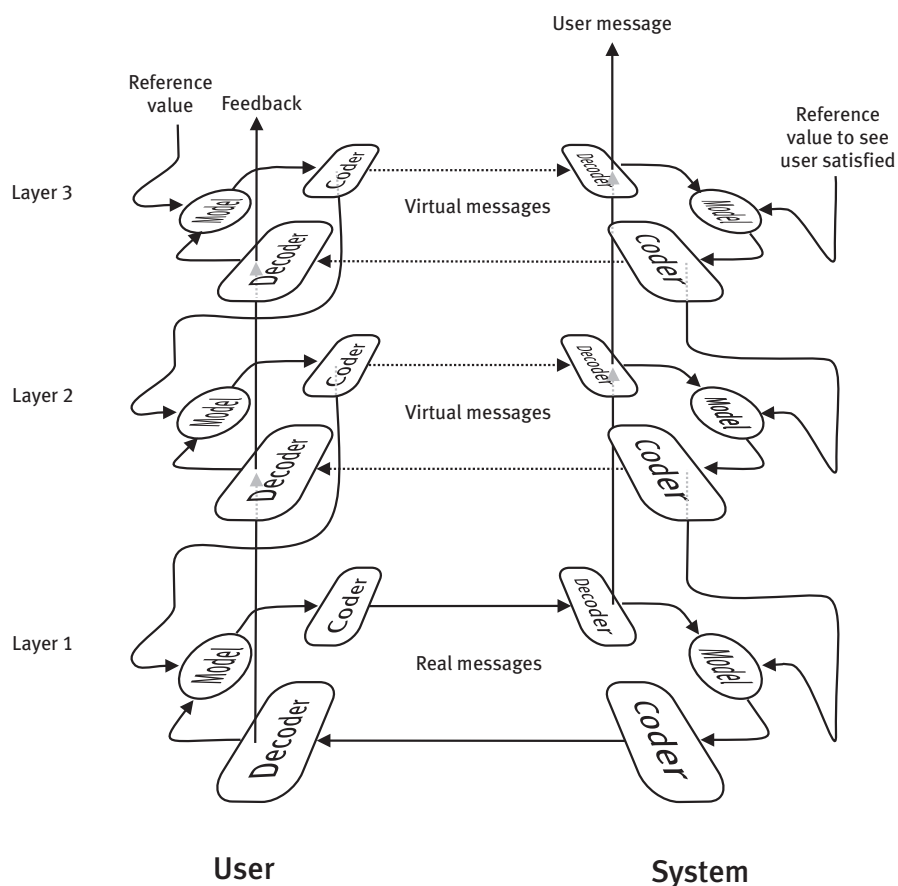


Figure 1.2: Layered user-system interaction. Via successive layers users send user messages to adjust their perception of the system state according to their reference value.

In LPT messages seem to pass from the user to the system and back within the same perceptual-control loop, but in fact all messages pass down from the originating sender to the bottom-level layer and from that layer up at the receiving chain to the receiver that operates in the same layer as the originating sender (Figure 1.2). As already mentioned,

users create sub-intentions as their intentions can not directly be mapped in physical actions. This creates a stack of perceptual-control loops with less abstract intentions when going downwards. In a Decoder-Model-Coder arrangement (Farrell et al., 1999), a Coder transforms a message into reference values for lower-level control loops. Once, the reference values are met, at the lower-level layers, the message is sent successfully. On the receiver side, a Decoder transforms the received lower-level messages back into the original message sent. Therefore, the message exchange on the lowest-level layer, the physical layer, is considered to be real, while on all higher-level layers it is virtual. The function of the Model in this arrangement is to compare the perceived system state with the reference value. If it deviates, the Model sends a message, which the Coder transforms. Instead of setting the reference value of one lower-level control loop, the Coder can also set the reference value of multiple lower-level control loops. In that case, a single higher-level message is supported by more than one lower-level control loop. The opposite can also happen, when multiple higher-level control loops appeal to one lower-level control loop for their communication. These constructions are called diviplexing and multiplexing, respectively (Taylor, 1988b; Taylor & Waugh, 2000). User-system interaction should, therefore, not be seen as two linear sequences, but as networks where multiple higher-level loops are linked with multiple lower-level loops.

1.3 Component-based software engineering

Component-Based Software Engineering (CBSE) is a sub-discipline of software engineering, which is primarily concerned with the following three functions: development of software from pre-produced parts; the ability to re-use those parts in other applications; and easy maintenance and customisation of those parts to produce new functions and features (Heineman & Councill, 2001). LPT fits into this concept well. Although the theory was not explicitly set up for it, some scientists argued that in essence this is what the theory was intended to accomplish (NATO, 2001). CBSE is not new; McIlroy (1979) already introduced the concept at the first conference on software engineering in 1968. He pointed at the inefficiency of software engineering when similar software parts had to be rewritten for different applications. Therefore, he envisioned a catalogue of software components where developers could choose and re-use. According to Myers (1998), the Andrew project (Palay et al., 1988) was the first to demonstrate that CBSE can also be applied to user interfaces. The idea behind the engineering concept is that components can easily be re-used in other systems since they are autonomous units, free of the context in which they are deployed. Components provide an external interface to their functionality and hide all details about internal constructs that go into providing this functionality. The promise of CBSE is reduced development cost and time (Aykin, 1994) since ready-made and self-made components can be used and re-used.

1.3.1 Component-based software development

The development organisation for CBSE differs from the ‘traditional’ engineering process that focused on developing each application from scratch. The component-based approach links many application development projects with four concurrent processes that are responsible for creation, management, support and re-use of the components (Figure 1.3). A creation process identifies and provides re-usable assets aimed at the needs of the re-users —developers using the existing components to build new applications. A supporting process maintains and distributes from the re-usable asset collection. The actual application development takes place in a re-use process. This process also is responsible for collecting and analysing the end-users’ needs. Since the creation and deployment of components are disconnected, a framework in which the usability of the individual component can already be assessed after its creation is of course welcome. Again, the validity of such an assessment depends on the extent to which the usability of a component can be isolated from the context of the application.

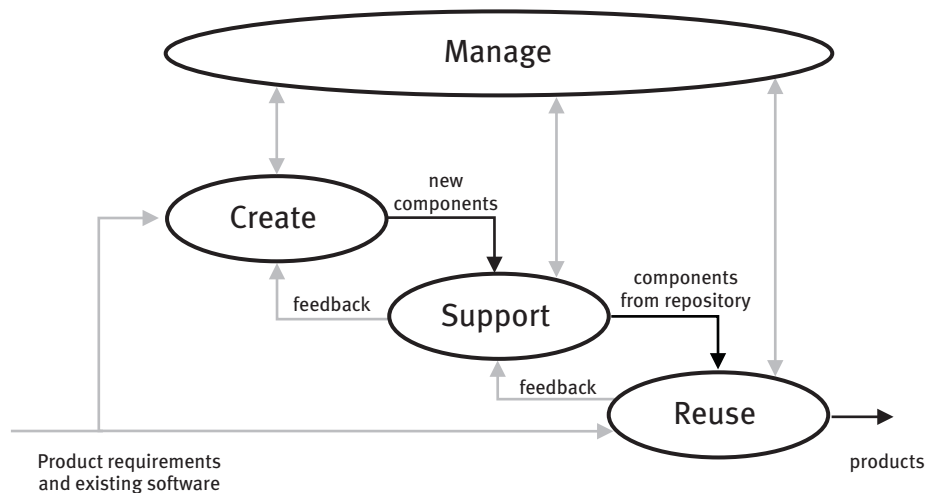


Figure 1.3: Four concurrent processes are involved in component-based software engineering (Jacobson, Griss, & Jonsson, 1997).

1.3.2 Software architecture

Whether control loops can be identified and component-based usability testing might be feasible in a particular interactive system depends on the extent to which the underlying software organisation supports the notion of control loops. In principle, all interactive systems can be described within LPT control loops since the complete system can be regarded as the control object of one large control loop. However, some software organisations may allow multiple control loops, or even joined control loops. Several software models have

been suggested to organise the components of interactive systems. These models guide developers of new systems and help maintainers to understand the organisation of existing systems. Models are classified according to the following three levels of abstraction that they address: the conceptual, the architectural and the realisation level (Duce, Gomes, Hopgood, & Lee, 1990). *Conceptual* models as the Language model (Foley & Dam, 1983), the Seeheim model (Green, 1983), the Arch-Slinky model (The UIMS Tool developers workshop, 1992) and the Lisboa model (Duce et al., 1990) provide no insight into the support of control loops. They focus on the identification of the different functionalities that should be presented in an interactive system. On the other hand, *realisation* models are too detailed because they focus on how components are implemented (e. g. imperative or object oriented program environment) and how the communication between them is established (e. g. components' interface definition). Support for control loops is best studied on the *architectural* level. Here, models focus on how components are combined to form a system. A group of models proposed on the architectural level are the so-called agent-based models. Agent-based models structure an interactive system as a collection of agents. An agent has a state, possesses an expertise, and is capable of initiating and reacting to events (Coutaz, Nigay, & Salber, 1996).

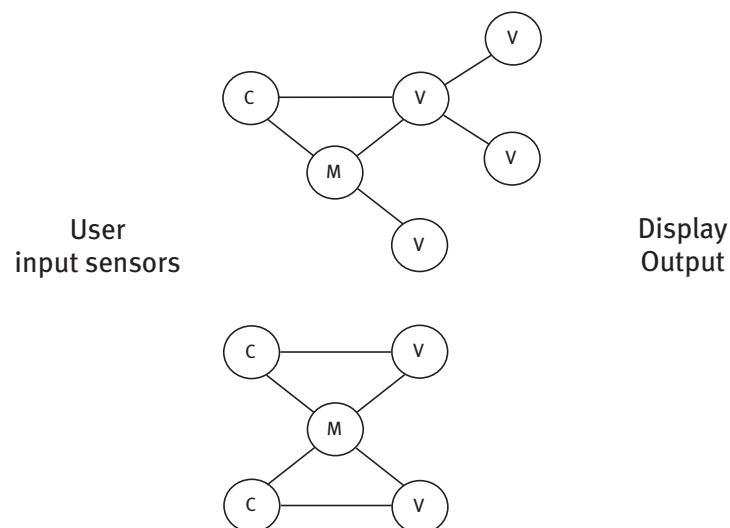


Figure 1.4: Two examples of an agent according to the Model-View-Controller (MVC) model (Krasner & Pope, 1988).

The Model-View-Controller (MVC) model (Krasner & Pope, 1988) (Figure 1.4) is an agent-based model that was first applied in the user interface of the object-oriented programming environment Smalltalk-80. Each MVC agent presents at least one control loop because users interact with a Controller to control the content of the Model, which is presented by a View. The Model links the agent with the underlying application domain. The relation between the agents, and therefore the relation between the control loops, is not specified in the MVC model. Each View may be thought of as being closely associated

with a Controller. Controllers and Views have exactly one Model, but a Model can have one or several Views and Controllers associated with it. Views can also contain subviews or be contained within superviews. Display messages, for instance, are passed from the top-level View through to the subviews. In a standard interaction cycle users perform some input action and the Controller notifies the Model to change itself accordingly. After this the Model broadcasts to its dependent Views and Controllers that it has changed. Views can then ask the Model about its new state, and update their display if necessary. Controllers may change their method of interaction depending on the new state of the Model. Controllers may also contact the View directly if the Model's state is not changed, but a change in the View is requested (e. g. zooming into a graph).

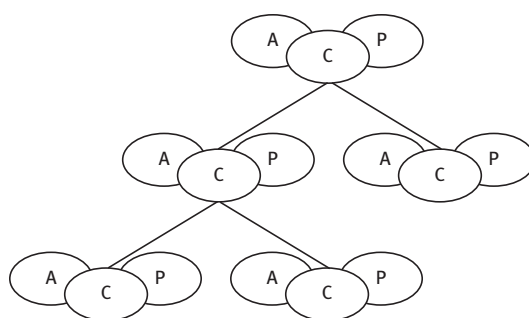


Figure 1.5: Example of five agents according to the Presentation-Abstraction-Control (PAC) model (Coutaz, 1987).

PAC (Presentation, Abstraction, Control) (Coutaz, 1987) is another agent-based model (Figure 1.5). This model also has a three-way division, but orders the agents in a hierarchical fashion. Contrary to MVC, the input and output is handled by one part, a Presenter. An Abstraction part maintains the data model that underlies the agent, and provides functionality that operates on this data. A Control part connects the Presentation and Abstraction part, and provides communication with other agents. In PAC, no direct communication between the Presentation and the Abstraction is allowed. All internal as well as external communication is directed through the Control part. The hierarchical structure of the external communication reflects LPT's idea of diviplexing, but not of multiplexing. For instance, the Control of a so-called cement agent combines user actions distributed over multiple lower agents into higher abstraction (e. g. command) (Nigay & Coutaz, 1991). Furthermore, the Control of a higher agent can also maintain the consistency between multiple presentations of the same concept in lower agents; any user action on one presentation is reported to the other agents, which presents the same concept, via the higher level agent. A hybrid model between the conceptual Arch-Slinky model and architectural PAC model has been proposed in the shape of the PAC-Amodeus model (Nigay & Coutaz, 1991, 1993). This model was suggested to handle multimodal user interfaces.

Of the three agent-based models presented here, the CNUCE model (Paternò, 2000) ex-

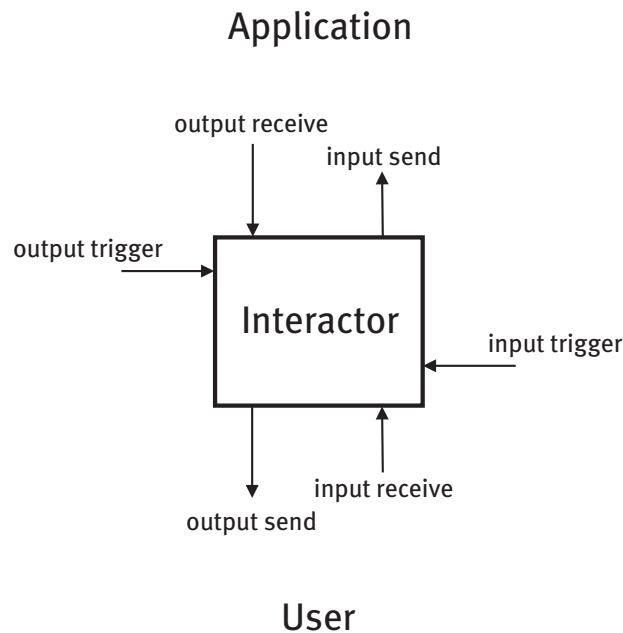


Figure 1.6: An agent according to the CNUCE model (Paternò, 2000).

presses the ideas of LPT in most detail. The model supports both multiplexing and diviplexing. In the model, agents, called interactors, can be seen as black boxes communicating bi-directionally with the user and application by channels called gates, of which there are six different types (Figure 1.6). Each interactor may receive input and output from other interactors and not only from the user or the application. Connecting these interactors creates a user interface. The input-send gate of one or more interactors can be connected to the input-receive gate of one or more other interactors, which means diviplexing and multiplexing on the receiver side in terms of LPT. Diviplexing and multiplexing on the sender side also is possible within the CNUCE model as the output-send gate of one or more interactors can be connected to the output-receive gate of one or more other interactors.

1.4 Usability evaluation

Usability is 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 (ISO, 1998). Within LPT, the focus on usability is the focus on control. Therefore, the ISO usability definition at the beginning of this section can be reformulated as follows. Usability is the extent to which a product can be controlled by specified users to match a reference value with effectiveness, efficiency and satisfaction in a specified context of use. These *ultimate criteria* of usability may be simple and direct, however, they can only really be measured

when users use the product in real life. As a result, other attributes are established, called *actual criteria*, which are more easily measured (e.g. not in real life, or without users) and are also believed to be effective predictors of the ultimate criteria (Hartson, Andre, & Williges, 2001).

Several methods have been suggested to evaluate the usability of a product. They can be divided into analytic and empirical evaluation methods. In *analytic* evaluations, people reason about the usability without involving users. The reasoning can be based on theoretical models. GOMS (Card, Moran, & Newell, 1983) and the Cognitive Walkthrough (Polson, Lewis, Rieman, & Wharton, 1992) are examples of theoretical model-based methods that simulate the cognitive and sometimes physical steps needed to fulfill a task. Analyses of the dialogue context and interaction points (Rauterberg, 1995) are methods that quantify usability based on product features. On the other hand, Heuristic Evaluation (e.g. Nielsen & Molich, 1990) is not based on a theoretical model, but on a set of usability principles (the ‘heuristics’).

Usability testing, questionnaires and field studies are examples of *empirical* evaluation methods. Empirical methods, for their part, can be classified according to the nature of the data they rely on, qualitative or quantitative. *Qualitative* analyses are common in naturalistic research traditions because they focus on the plausibility and soundness of the interpretation of observations; i.e. examining interview transcripts, thinking-aloud protocols or video recordings. Characteristic of these analyses is their subjective and often time-consuming nature. *Quantitative* analyses are more related to cognitive and behavioural traditions (Sanderson & Fisher, 1997) because they focus on replicability and generality, and call on statistical analyses.

The formative versus summative classification, often made for evaluation methods (Hartson et al., 2001), is somewhat problematic in the case of an empirical component-based usability test. *Formative* evaluation is done during the construction of a product to find usability obstacles for improvement and has a more explorative and qualitative nature. However, what should be considered as the formative period in CBSE, the creation or the re-use process? On the other hand, *summative* evaluation is used to assess or compare the level of usability achieved in an entire product when finished. Summative evaluation is generally regarded as requiring formal experimental design, including statistical analyses and is often used to compare design factors in a way that can contribute to the accumulated knowledge within the research field. However, instead of the entire product, multiple versions of a component can be compared. In such a case, the evaluation is not intended to extend the knowledge of the research field, but to select the most usable version of a component that can be offered for re-use.

1.5 The research objectives

1.5.1 An empirical component-based testing framework with a theoretical background

LPT can support designers during the design and evaluation of a product by offering a theoretical background. Cases are reported where designs are analytically evaluated according to guidelines derived from LPT (Edwards & Sinclair, 2000; Farrell et al., 1999; Farrell & Semprie, 1997) and consequently, empirical testing has been done to show that these guidelines can produce more usable products (Eggen, Haakma, & Westerink, 1996; Haakma, 1998). The benefit of LPT, compared to the original GOMS method (Card et al., 1983), is that it allows for product analysis of the product's initial use phase, and not only of its long-term use phase (Eggen, Haakma, & Westerink, 1993). However, another member of the GOMS family, NGOMSL (John & Kieras, 1996), can predict the learning time for both procedural knowledge and memorizing chunks of declarative information. Still, all members of the GOMS family only focus on error-free task execution and do not consider the system's support to prevent or to overcome user errors, which is especially crucial in the initial use phase. The cognitive model on which GOMS but also the Cognitive Walkthrough is based does not consider layered user-system interaction. Although the cognitive model has a goal hierarchy, it considers the system only at the lowest-level layer. The system is reduced in more or less the same way as the cognitive process is reduced to the keystroke level by the Keystroke-Level Model (KLM) (Card, Moran, & Newell, 1980), which is the simplest member of the GOMS family. Each layer in LPT includes a system part, which the cognitive model lacks. This makes it possible to trace back usability problems to higher-level system layers too, instead of only the direct relation between unclear labels or icons and the lowest-level system layer.

The research, reported in this thesis, aims to establish an empirical test method in which the usability of a component can be assessed. Although, LPT is suggested as an approach to test components empirically (Haakma, 1998; Hilbert & Redmiles, 2000; Taylor, 1988b), it has not yet been applied. Therefore, this is seen as a first attempt to establish a formal observation method on whether users successfully controlled their perception in each of the different layers. Again, the benefit is that such a method points out troubling components to designers, and does not leave them in the dark about what to change in the case of disappointing overall performance (Olsen & Halversen, 1988).

The research approach is to evaluate a component within the context of a complete user interface. Attempts to empirically study a component without other components would face a very artificial situation especially for higher-level components. In an effort to study a Internet web-site without a browser, testers would have to offer the users another way to navigate through it, which again would mean creating some kind of browser after all.

1.5.2 Is the whole more than the sum of its parts?

With the component-based usability testing framework established, the question is addressed whether the component's assessed usability holds outside the context of the user interface in which it was tested. LPT claims that layers can be designed almost independently of each other. Control loops are said to be unaffected when lower-level protocols are replaced as long as they provide the same message services to the layer above it (Taylor, 1988a, 1988b, 1993; Taylor et al., 1984). If this were the case, the overall usability of new user interfaces could be solely based on the usability of the separate ready-made components. On the other hand, Hertzum (2000) suggests that software re-use can cause conceptual mismatches. The same concept may be used in several components, but it may not mean the exact same thing. This basically means that components influence each other's usability. If so, designers should control possible negative impacts, while re-users of the components should avoid compositions that amplify them. Furthermore, when looking at usability test results, testers should be alert to the possibility that the root of a usability problem may be located partly outside the component.

1.6 Overview thesis

The organisation of this thesis chronologically follows the research conducted on the main theme —Is usability compositional? The research starts with the assessment of the component's individual usability. Chapter 2 describes an explorative experiment conducted on a fictitious user interface. The experiment is a first attempt to record the interaction on multiple layers and to describe the elements of the user interface involved. Based on the results and the experience, an objective component-specific usability measure is suggested of the users' effort to control their perception of a component. In chapter 3, a complete testing framework is worked out in which the usability of individual components of a user interface can be tested. Three component-specific usability measures are proposed: an objective performance measure, a perceived ease-of-use measure and a satisfaction measure. The objective performance measure is derived from the message exchange between the components recorded in a log file, whereas the other measures are obtained through a questionnaire. The chapter also describes how a component-specific usability test can be performed when only one or when multiple component versions are available. Chapter 4 describes an experiment conducted to evaluate the suggested testing framework. Eight prototypes of a mobile telephone were subjected to a usability test. The power of the component-specific measures is compared to overall measures such as task time, number of keystroke, ease-of-use and satisfaction related to the entire mobile telephone.

The second part of the research is concerned with the related question whether the usability of a component is independent of the other components in the user interface? Two factors are studied that might cause components to influence one another, i. e. consistency and mental effort. Chapter 5 describes three experiments in which the effect of inconsistency is

examined between components on the same or a different layer, and between components and the application domain. Prototypes of a room thermostat, a web-enabled TV set, a microwave and an alarm clock radio served as evaluation objects in these experiments. An experiment on the impact of mental effort is presented in chapter 6. The hypothesis is tested that the mental effort, related to the interaction of lower-level component, can affect the users' strategy to control a higher-level component. Two calculator prototypes were put to the test in this experiment.

Chapter 7 reflects on the conclusions drawn in the prior chapters and discusses the theoretical and practical implications of the results, limitations of the research, and possible further research.

Chapter 2

An explorative study on control and the comprehension of user interface components

2.1 Introduction

Overall usability measures suit a world where the entire user interface is considered as a single entity. However, in a world where a user interface is considered as an assembly of different components, these measures seem too global. Component-specific usability measures would then be more appropriate. These kinds of measures may distinguish components that limit the overall usability of a user interface and they may also shed light on the interaction between the components of a user interface. Until now, no such measures have been developed. Therefore, a first step forward requires an explorative study, where observation of users' control could help to generate ideas for potential empirical and component-specific measures. When observations are recorded in a log file, measures can be derived and studied at a later moment. However, what exactly should be recorded? The Layered Protocol Theory (LPT) may give some directions here. However, the description of the state-of-the-art on LPT in chapter 1 does not present the exact recipe for describing the interaction or the user interface architecture. This means that attention needs to be paid to operationalising LPT's concepts as well.

2.1.1 Description message exchange between components

In LPT, control is established by the exchange of messages between the user and the system on several layers. Therefore, recording the message exchange implies recording both the user and the system behaviour on several layers. Whereas the messages sent and received by the users on higher-level layers are not directly observable, the messages sent and received by the system are unambiguously and known.

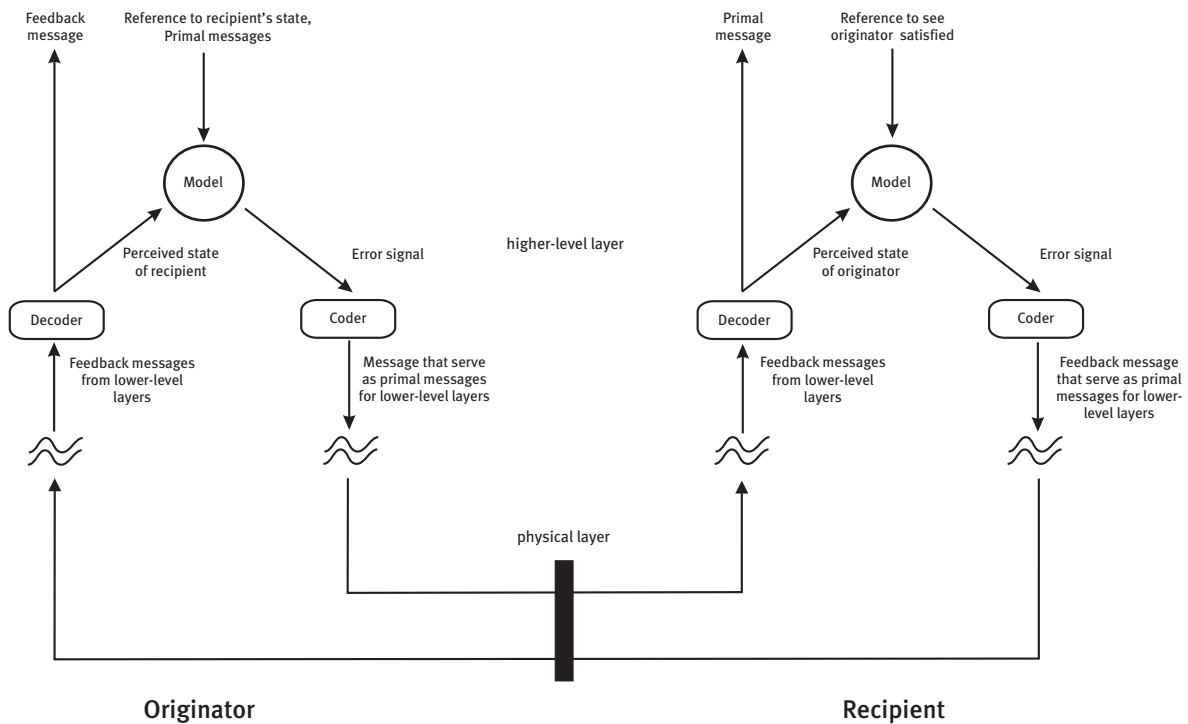


Figure 2.1: The relation between an originator and recipient protocol node according to the Decoder-Model-Coder arrangement.

Different kinds of messages can be distinguished in the message exchange within the perception-control loops that operate in each of the layers. To include E-feedback messages and to focus more on the user-system interaction, a new arrangement, the so-called Decoder-Core-Coders arrangement (Figure 2.2), is introduced here, which is a slight variation on the Decoder-Model-Coder arrangement (Figure 2.1) mentioned in the previous chapter (Figure 1.2). The Model in a receiving protocol node performs an evaluation function similar to the one that takes place at the user side of Decoder-Core-Coders arrangement. The Model compares the system reference value, which is always *the originator is satisfied* in a co-operative dialogue, with the perceived satisfaction of the user. Receiving a user message is always seen as an indication that the user is not satisfied, which makes the Model in a receiver protocol node a more or less artificial construct. On the other hand, the Core is simply regarded as the element that acts on the user's messages and sends both interpretation and expectation feedback to the user. The message the Core receives from higher-level layers is not a reference value, but a higher-level feedback message, which has to be passed on to the user. The Core can only send feedback directly at the physical layer. In all the other layers, a Coder has to transform a feedback message into one or more lower-level messages. In contrast to the Decoder-Model-Coder arrangement, two Coders are distinguished. The I-Coder transforms the I-feedback and the E-Coder the

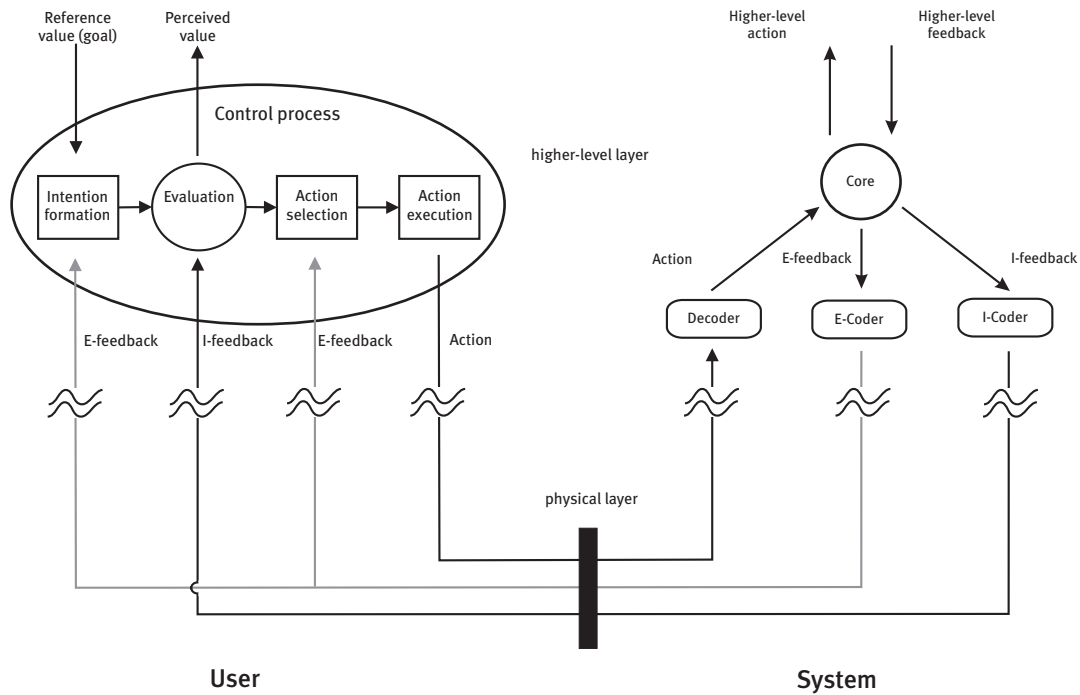


Figure 2.2: Perception-control loop in which a user controls a system component according to the Decoder-Core-Coders arrangement.

E-feedback. The Decoder performs the opposite function; it transforms lower-level messages into one user message at that level and sends it to the Core. This again differs from the Decoder-Model-Coder arrangement, where the Decoder sends messages to the Model and also to higher-level protocol nodes. Instead, the Core sends messages upwards since only it can interpret the received messages in the light of the previous received messages and its current state.

2.1.2 Objectives

The aim of this explorative experiment is to apply the Decoder-Core-Coders arrangement and to search for a component-specific usability measure that can be constructed from the recorded messages sent and received on the system side of the interaction. Furthermore, the aim also was to form an idea of how likely it is that a component's usability influences the users' control of other components.

2.2 Method

The experiment was conducted under the control of a PC application written in Visual Basic 5. In the experiment, the subjects had to execute a task with a fictitious user interface. Systematically manipulating the subjects' initial knowledge of the user interface created usability variations in this experiment.

2.2.1 Prototype

A user interface was designed with components on different layers. This user interface presented an abstraction of a dialogue that is essential in multi-layered user interfaces. In this dialogue the effect of a user action depends on the current system mode. For example in time setting, users first have to select the hour-digits (select the right mode) before they can change the hours of a digital watch. In the time-setting example, three components can be distinguished: the physical buttons, a component to select the digits of the current time (hours, minutes, or seconds), and third a component to set the digits. Similar components were used in the user interface studied in this experiment.

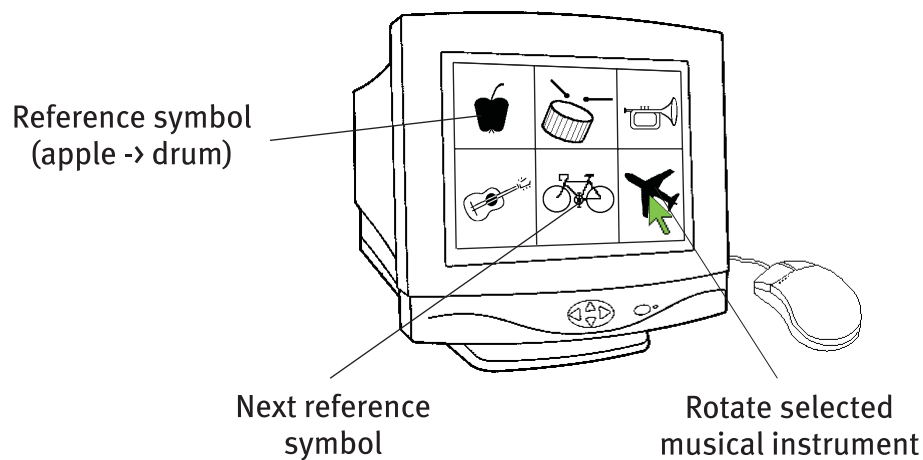


Figure 2.3: Computer screen with the fictitious user interface.

The subjects were confronted with a fictitious interface on the computer screen. The user interface (Figure 2.3) used symbols in which the signifier (the form which the sign takes) had no direct relation with the signified (the mental concept it represents). In terms of semiotics (Chandler, 2002), this is a truly symbolic relation in which the signifier does not resemble the signified but which is fundamentally arbitrary. In other words, this relationship had to be learnt. Instead of setting the current time, the subjects were asked to rotate the symbol of a specific musical instrument. The system consisted of two buttons

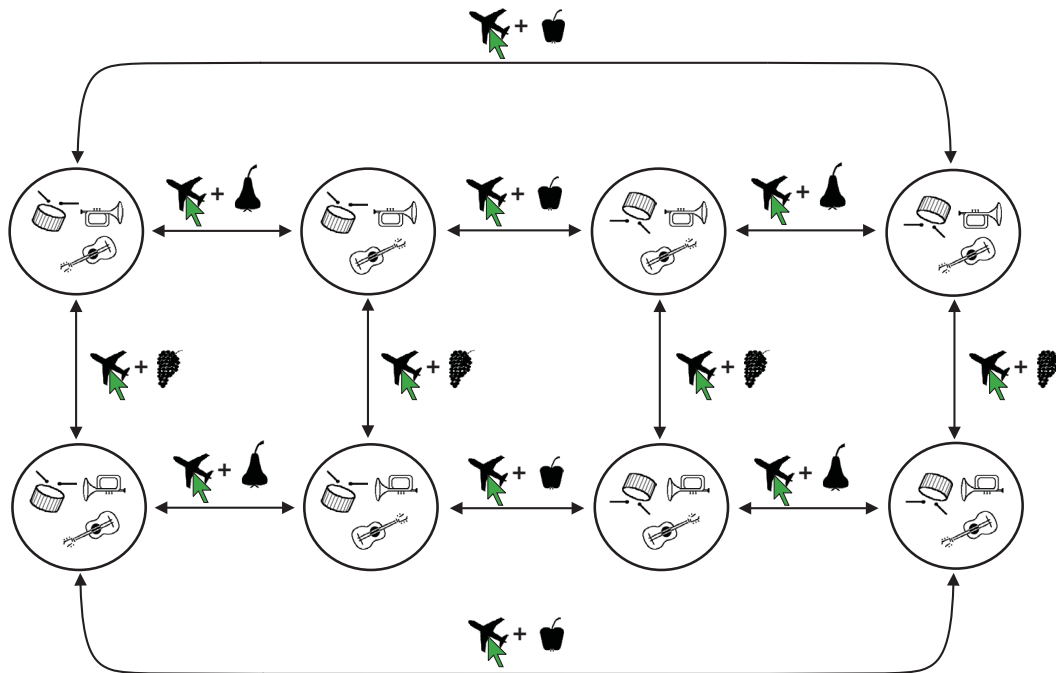


Figure 2.4: State diagram of the Rotate component. A click on aeroplane symbol, in combination with the visible fruit symbol, triggered a particular state transition.

components (Bike and Aeroplane), a Select component to select a musical instrument, and a Rotate component to set the orientation of the musical instruments (Figure 2.6).

To rotate a musical instrument, subjects had to click on an aeroplane symbol. Which musical instrument would rotate depended on a fruit symbol visible (Figure 2.4). If the apple was visible, the drum would rotate. If the pear was visible, the guitar would rotate. And if grapes were visible, the trumpet would rotate.

To change the visible fruit symbol, subjects had to click on a bike symbol. Each click on the bike symbol changed the fruit symbol in turn (Figure 2.5). For example, a click on the bike symbol in Figure 2.3 would change the apple into a pear symbol. However, a click on the aeroplane symbol would rotate the drum. If the subjects wanted to rotate the trumpet in Figure 2.3, they first had to change the apple into grapes by clicking twice on the bike and finally once on the aeroplane.

The structure of the message exchange between the components is given in Figure 2.6. The Rotate component was on the highest-level layer of the interface structure. The I-Coder of this component presented the musical instruments in a normal or upside-down orientation according to their current status. The E-Coder split the E-feedback into a message for the Aeroplane Button components that represented the rotate 'button', and a message for the Select component that was engaged in selecting the musical instrument. When the Decoder of the Rotate component received an *Aeroplane button pressed* message, it combined this

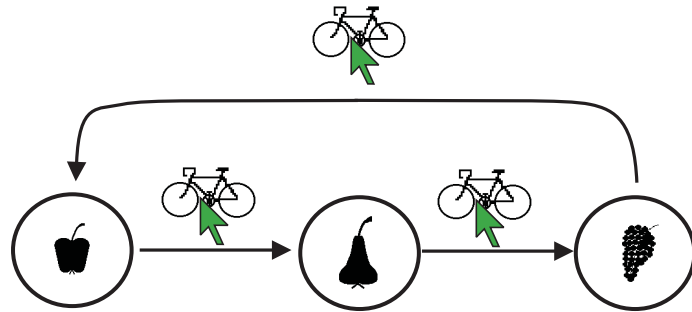


Figure 2.5: State diagram of the Select component. A click on the bike symbol triggered a state transition.

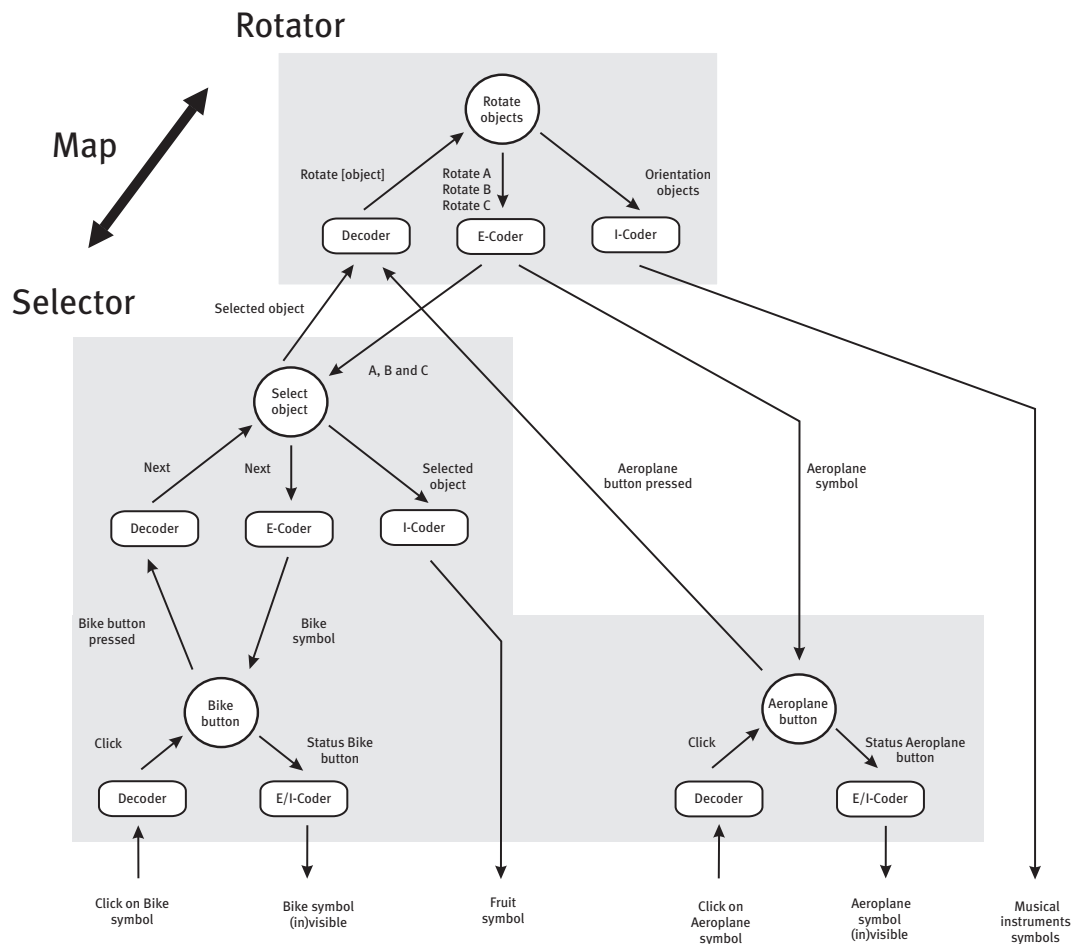


Figure 2.6: Compositional structure of the fictitious user interface in which the message exchange is presented by the arrows. The grey areas refer to the different parts of knowledge given in the training.

trigger with its information about the selected object into a Rotate message for a specific object.

The Select component informed the Rotate component of the currently selected object each time the subjects selected another object. Presentation of the Select component's E-feedback was delegated to the Bike Button component, which in return informed the component when the subjects click on the Bike symbol. The I-feedback of the Select component was presented as a fruit symbol on the screen.

The E-feedback of the button components was based on the rule that 'clickable' symbols were all means of transport, e. g. the bike or the aeroplane. To indicate to the subjects that they had clicked on a clickable symbol, the symbol disappeared for half a second from the screen. Both the Bike Button and Aeroplane Button components did this. After the subjects clicked on a bike symbol, the Bike Button component sent a *Bike button pressed* message to the Select component. Likewise, after the subjects clicked on the aeroplane symbol, the Rotate Button component sent an *Aeroplane button pressed* message to the Rotate component.

2.2.2 Experimental design

The subjects' initial knowledge, obtained in a training, and the experience obtained when using the system, was expected to influence the subjects' control on the different components (Figure 2.7). In the training, prior to the task, the subjects received either relevant or irrelevant information about the system response to particular user actions. The difference in foreknowledge was assumed to affect the overall performance of the interactive tasks, as it did in other studies (e. g. Kieras & Bovair, 1984; Halasz & Moran, 1983). The subjects were instructed to perform the same task nine times. The expectation was that over the trails the subjects would extend their knowledge, which consequently improved their control.

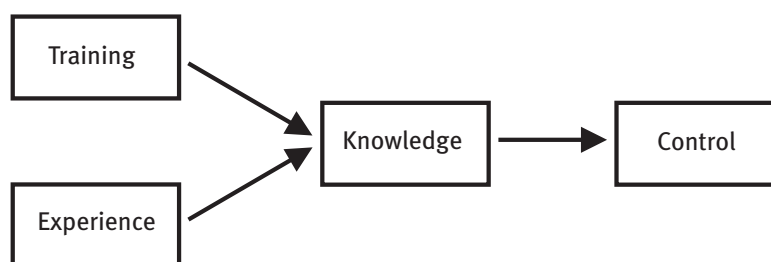


Figure 2.7: Conceptual model of the experiment: on the left the independent variables, *training* and *experience*; in the middle the intermediate variable *knowledge*; and on the right the dependent variable *control*.

The subjects were assigned to one of eight instruction sets (Table 2.1) in the training. The following three parts of information about the fictitious user interface were given or

Table 2.1: Knowledge about the fictitious user interface given to or withheld in the eight instruction sets.

Knowledge	Instruction sets							
	I	II	III	IV	V	VI	VII	VIII
Rotator	0	0	0	0	+	+	+	+
Map	0	0	+	+	0	0	+	+
Selector	0	+	0	+	0	+	0	+

Note. + = Relevant knowledge; 0 = Irrelevant knowledge.

Ten subjects were assigned to each instruction set.

withheld in each instruction set: first, the Rotator knowledge, the knowledge that a click on an aeroplane would lead to the rotation of the selected musical instrument; second, the Map knowledge, the knowledge of which piece of fruit refers to which musical instrument; third, the Selector knowledge, the knowledge that a click on a bike would lead to the selection of the next fruit symbol and that clickable symbols were means of transportation. Consequently, the Rotator knowledge related to the higher-level layer of the compositional structure, the Selector knowledge to the lower-level layer, and the Map knowledge served as an intermediary between those layers.

2.2.3 Training

In the training, the subjects were simultaneously give six examples of two images of a computer system on the PC screen (which means twelve computer systems were displayed on the PC screen at the same time). Each example presented how the computer would react to a click on a symbol. A single example consisted of an image of two computers, one below the other, a arrow connecting them, and an illustration of a hand pressing a mouse button (Figure 2.8A). The upper computer presented the system state before the action. A mouse arrow indicated which symbol was about to be clicked on. The lower computer showed the system state after the click action. The arrow and the hand were put into the example, so the subjects would interpret the example as a transition between system states.

Part A of Figure 2.8 is an example that the subjects with instruction set VI and VIII had to study. They had to notice that a click on a bike would change the pear into grapes. The subjects had to study two more examples of a *next* action to understand all the state transitions of the Select component (Figure 2.5). In the instruction set II, IV, VI and VIII this meant an example of a transition of grapes into an apple and in the other example an apple into a pear. Part B of Figure 2.8 presents the signified element of the symbols in these examples. The subjects with another instruction set also studied these three change

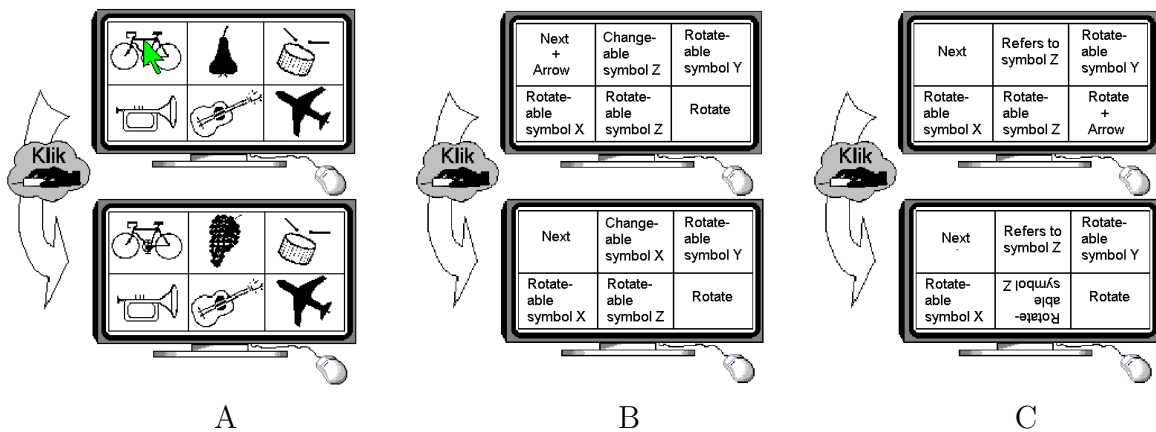













































































































































Figure 2.8: One of the six examples. Left (A) a real example of a *next* action that the subjects with instruction set VI and VIII had to study. In the middle (B), the signified element of the symbols in a *next* example. Right (C) the signified element of the symbols in a *rotate* action example.

examples; however, dummy symbols (symbols of animals and instead of a bike a kite) were used to withhold the Selector knowledge them from. Table 2.2 shows all the symbols that were used in the eight conditions. Besides the three *next* examples, subjects had to study three *rotate* examples. Part C of Figure 2.8 presents the signified element of the symbols in this example type. The actual rotate examples the subjects had to study can again be constructed with the help of Table 2.2. With the rotate examples, subjects received or were deprived of the Rotator and Map knowledge, and a part of the Selector knowledge that told them that only means of transport were clickable symbols.

When subjects felt that they had thoroughly studied the examples, their knowledge was tested. In the test, the subjects received (one by one) the six examples in a random order in which only the upper part of the example was filled out. The task of the subjects was to complete the screen of the computer below, i. e. to predict the reaction of the system to the action presented in the upper computer. To do this, the subjects had to pick a symbol from a set and place it in the image of the lower computer screen, which was left blank. After the subjects had completed the six examples, the application checked whether their answers were correct and showed the subjects the result. The application gave the number of rotation errors and the number of displacement errors. The six before and after situations, as the subjects had filled them out, were all presented on one screen at the same time. The application indicated with a red cross which symbols were wrong or were wrongly orientated. The subjects could study their results. After this the subjects could again study the six examples and do the test again. The subjects finished the training when they were able to pass the test without making any errors.

To make sure that the subjects studied the symbols and not the location of symbols, the places of the symbol in the upper computer system were randomly chosen each time that

Table 2.2: Symbols used in the 6 examples of the instruction sets.

Knowledge	Signified	Instruction sets (signifiers)							
		I	II	III	IV	V	VI	VII	VIII
Rotator /	1 Rotation symbols	Figures	Figures	Music	Music	Music	Music	Music	Music
Map	1.a Rotate-able symbol X			  	  	  	  	  	  
	1.b Rotate-able symbol Y			  	  	  	  	  	  
	1.c Rotate-able symbol Z			  	  	  	  	  	  
Map	2. Reference symbols	Tools	Tools	Fruit	Fruit	Tools	Tools	Fruit	Fruit
	2.a Refers to symbol X								
	2.b Refers to symbol Y								
	2.c Refers to symbol Z								
Selector	3. Clickable objects	Sky	Transport	Sky	Transport	Sky	Transport	Sky	Transport
Rotator	3.a Rotate	 	 	 	 	 	 	 	 
Selector	3.b Next								
Selector	4. Changeable objects	Animals	Fruit	Animals	Fruit	Animals	Fruit	Animals	Fruit
	4.a Changeable symbol X	 	 	 	 	 	 	 	 
	4.b Changeable symbol Y								
	4.c Changeable symbol Z								

Note. In instruction set I, only dummy symbols were used, whereas in instruction set VIII, only symbols were used from the fictitious user interface. Symbols in the column with a label printed in italics were linked with relevant knowledge about the fictitious user interface.

the application presented the examples. Next, the position of the six examples on the screen was also randomly chosen, thus avoiding the idea that the order of the examples would in some way help to predict the reaction of an action. A pilot study of the experiment had shown that subjects suffer from concentration problems, if the learning task took too long. This appeared to affect their performance on other tasks. Therefore, the subjects who had not passed the test after 20 minutes received an oral explanation of the written instruction about how to study the examples. The subjects who still had not successfully passed the test after 40 minutes were given help in studying the examples until they were able to construct the examples successfully.

2.2.4 Hypotheses

In the experiment both the intermediate variable *knowledge* and the dependent variable *control* were measured. The knowledge measuring served as a confirmation check, to justify the interpretation that the knowledge the subjects obtained as they performed the rotation tasks influenced their control. Furthermore, the knowledge about a component, which they obtained in the training, was expected to influence their control on the specific component as well. The last expectation was that the Rotator, Map and Selector information provided in the training would not cause interaction effects in the control of a component. In other words, the components' usability, presented here as the subjects' initial knowledge, did not affect each other.

Measuring knowledge

The subjects' knowledge about the system was measured before and after the subjects performed all the rotation tasks. The measuring was done in a similar way to that in the training. However, subjects were asked to complete eight action-reaction combinations and this time the symbols were taken from the fictitious user interface (similar to instruction set VIII in Table 2.2). Of the eight images, two images presented a click on an aeroplane, two on a bike, two on a piece of fruit, and two on a musical instrument. The subjects were not informed about their performance in this test. The subjects were expected to complete more images correctly than before after they had performed all the rotation tasks.

Measuring control

The reported control measuring focused mainly on the user messages. At first, it might seem more obvious to analyse the relation between the reference value and the perceived value (i. e. I-feedback). A high correlation would indicate a highly accurate control over the system. However, these analyses would face two problems in this experiment. First, contrary to other analyses conducted within the framework of the Perception-Control Theory (e. g. Bourbon & Powers, 1999), both reference and perceived values were not continuous

variables that could be placed on one scale, but nominal (or categorical) variables; they were discrete system states for which the only direct comparison available was whether they were the same or not. Second, only the highest-level reference value was known since this was given in the task instruction (e.g. to see the trumpet rotated). Reference values on lower-level layers (e.g. to see the grapes) changed as the subjects set the right or wrong ones in an attempt to send the higher-level user messages. Still, in this experimental set-up, the subjects continued with sending messages as long as the high-level reference and the perceived values were not similar (Figure 2.9). The number of messages indicated the number of times the subjects went through a cycle of the control loop, which can be regarded as the amount of effort the subjects had to spend to change the perceived value into the reference value. Although sending a user message always took some effort, not all user messages resulted in an I-feedback message; for example, the system would not send I-feedback after a click on a musical instrument. Therefore, analysing the user messages instead of I-feedback seemed a more effective approach to studying the user effort invested in the control. Analysing the users' effort through the E-feedback messages was expected to give little insight since E-feedback was not the object of control, but merely guided the subjects in controlling it. An extra differentiation of the user messages per I-feedback, a kind of stimulus-response combination, was considered too detailed in this experiment since subjects were expected to understand the difference states of a component (I-feedback) equally well.

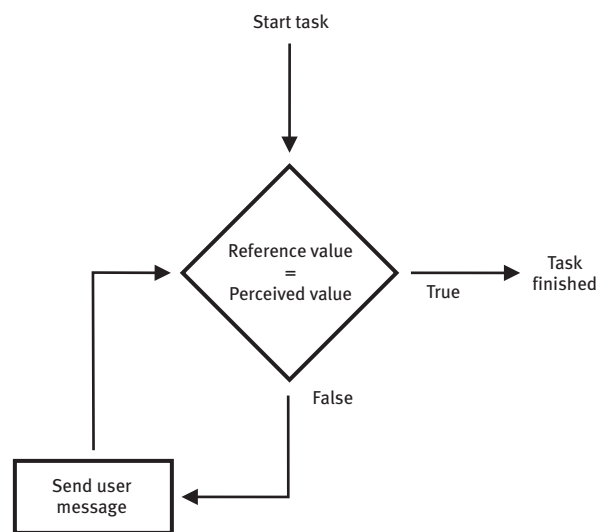


Figure 2.9: Process of the task execution within the experimental set-up.

The user messages led to one overall measure, three lower-level measures and three higher-level measures. The overall measure was the number of times the subjects clicked on a symbol. The three lower-level measures made a selection in the symbols clicked on. The three lower-level measures were the number of times the subjects click on the bike, the aeroplane, or on another symbol, respectively. The expectation was that effects for the

different instruction sets would be found in the corresponding lower-level measures (Figure 2.10). An effect for the Selector knowledge was expected to be found in the number of clicks on the bike and the other symbols because this knowledge concerned the control of the Select component and the rule about a clickable symbol. An effect for the Map knowledge was expected to be found in the number of clicks on both the bike and the aeroplane symbols because this knowledge concerned the selection of musical instruments. If this knowledge was missing, subjects had to click on both these symbols to obtain this knowledge by trail and error. Finally, an effect for the Rotator knowledge was expected to be found in the number of clicks on the aeroplane symbol because this knowledge concerned the triggering of a rotation.

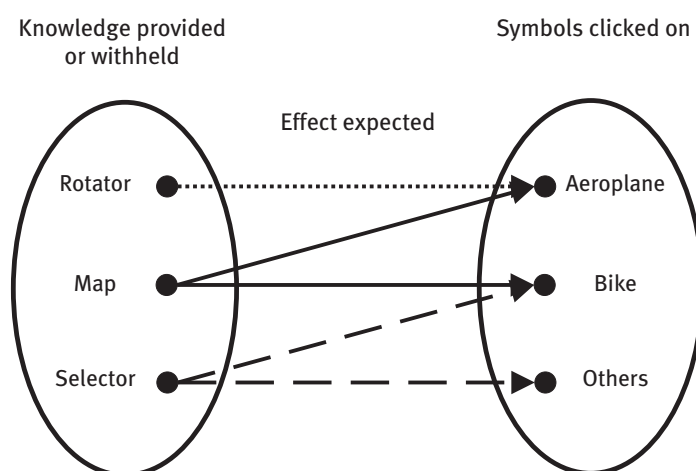


Figure 2.10: The effects expected for Rotator, Map and Selector knowledge in the three lower-level measures. A click on the aeroplane symbol would trigger a rotation. A click on the bike symbol would result in the selection of another musical instrument. A click on the other symbols (musical instrument and fruit) caused no system-state transition.

The three higher-level measures were the three Rotate [*object*] messages passed on by the Decoder to the Core of the Rotate component. The Decoder combined the *aeroplane button pressed* and the *selected object* message into this higher-level message. The three higher-level messages were the number of Rotate [*target*], Rotate [*target-1*] and Rotate [*target-2*] messages corresponding to the different states of the Select component (Figure 2.5). The ‘target’ was interpreted as the musical instrument the subjects were supposed to rotate, ‘target-1’ as the musical instrument selected before the target, and ‘target-2’ as the musical instrument that was selected before target-1 (or after the target). The expectation was that only effects for the Map and Rotator knowledge would be found in these measures.

Another potential measure of the subjects’ control on the component was the time between two user messages. The initial expectation was that this time could be attributed to evaluation of the E- and I-feedback. However, this measure suffered from pre-selection because the user messages preceding or succeeding the user messages of interest was also

exposed to the effect of the subjects training. This made it impossible to explain the effects found solely in terms of the user messages of interest. Therefore, reports about this measure are not given in the result section.

2.2.5 Procedure

Eighty students (54 male and 26 female) between 17 and 27 years old of the Eindhoven University of Technology participated in the experiment, for which they received NLG 15 (€ 6.81). In the experiment, the subjects went through the following five phases: welcome phase, training phase, pre-experimental test phase, task execution phase and post-experimental test phase. In the welcome phase, the subjects were brought into a test room of a usability laboratory. Here the test procedure was explained to them. In the training phase, the application showed the six action-reaction examples, which the subjects had to learn by heart. After completing the training, the subjects were tested for their knowledge of predicting the fictitious user interface reaction to an action. In the task execution phase, the subjects performed three rotation tasks in succession as quickly as possible. In a rotation task, the subjects had to operate a working version of the fictitious user interface. In each task, the subjects had to rotate a specific symbol. The three rotation tasks, each aimed at rotating a different musical instrument symbol. Before the subjects carried out the task, the system was reset to the same system state assuring that everyone had the same starting-point to begin with. To study the learnability (or experience), the subjects had to perform the rotation task nine times divided over three sessions. However, to prevent the subjects from establishing fixed action sequences without evaluating the I-feedback, the initial state of the fictitious user interface was different in each session. Between these sessions, the subjects performed a filler task; they operated a simulation program of an aquarium (Freudenthal, 1998; Vollmeyer, Burns, & Holyoak, 1996) for three minutes between the sessions. Here their task was to establish a specific population of sea animals by manipulation of environmental parameters. In the following phase, the post-experimental test phase, the subjects again were asked to complete eight after images, as they had done in the pre-experimental test phase.

2.3 Results

2.3.1 Control tests

Before the effects on the depended variable *control* were analysed, two analyses were conducted to confirm assumptions about the difficulty of the different instruction sets and about the effect experience had on the intermediate variable *knowledge*.

Table 2.3: Mean subject variables and global performance data.

Instr. set	Age	Male	Female	Training				Over all rotation tasks		
				Tests	Time ^a	Help		Help ^d	Time ^e	Clicks
						20< ^b	40< ^c			
I	21	5	5	2.9	22	4	0	4	6:03	166
II	23	5	5	3.9	28	6	2	1	4:23	102
III	20	9	1	3.1	29	7	1	2	3:54	87
IV	23	7	3	3.5	31	8	1	3	4:22	66
V	21	7	3	2.8	23	5	0	1	3:53	112
VI	23	6	4	3.6	29	7	1	1	4:11	92
VII	22	8	2	4.3	29	8	1	0	3:40	64
VIII	23	7	3	3.4	27	8	0	1	3:01	42

^aTime in minutes. ^bNumber of subjects that received an oral explanation of the instruction when a subject was not able to complete the training after 20 minutes. ^cNumber of subjects that received assistance in studying the examples when a subject was not able to complete the training after 40 minutes. ^dNumber of subjects that requested and received an oral explanation of the instruction on the rotation task. ^eTime in minutes : seconds.

Difficulty training

Table 2.3 shows the general results of the experiment. The assumption was that the different instruction sets were equally difficult to learn. If this was not the case, effects found in the control could also be explained as fatigue or motivational effect. However, Kruskal-Wallis tests on the number of times help was given, both after 20 minutes ($\chi^2 = 7.01$, $df = 7$; $p. = 0.483$) and 40 minutes ($\chi^2 = 4.98$, $df = 7$; $p. = 0.907$) did not reveal a significant effect for the instruction set. Therefore, it seems that none of the instruction sets was more difficult than the other ones to learn.

Knowledge gain by experience

The knowledge was measured to study whether it improved after the subjects performed the rotation tasks, which confirmed the expected effect of the independent variable *experience* on the intermediate variable *knowledge* (Figure 2.7). A doubly multivariate analysis of variance (doubly MANOVA) was conducted on the errors made in the knowledge tests on the system response of a click on the following symbols: the bike symbol, the aeroplane symbol, a fruit symbol, and a musical instrument symbol. The within-subjects variable was the Moment (2) of the test —before or after the task execution phase. The analysis found an effect for the Moment ($F(4,76) = 7.60$; $p. < 0.001$). Univariate analyses also

found this effect in the number of errors made in prediction of a click on the bike symbol ($F(1,79) = 7.82$; $p. = 0.006$), the aeroplane symbol ($F(1,79) = 7.09$; $p. = 0.009$), a fruit symbol ($F(1,79) = 24.42$; $p. < 0.001$), and a musical instrument ($F(1,79) = 22.11$; $p. < 0.001$).

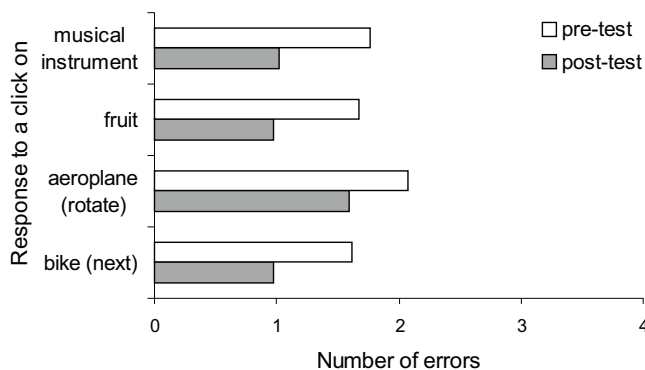


Figure 2.11: Number of errors made in the prediction of the system response on a click on a specific symbol. Maximum number of error a subject could make for the click on a symbol was 4 ($2 \times (1 \text{ rotation and } 1 \text{ displacement error})$).

The subjects made better predictions after they performed the rotation tasks than before (Figure 2.11) for all the system responses. This finding indicates that the subjects' knowledge of the components increased when the subjects interacted with them. Therefore, possible monotonous change over the three sessions in the dependent variable *control* was more likely caused by the increase of knowledge and was less likely solely caused by another factor such as the motivation to explore the user interface.

2.3.2 Overall performance

The overall measures were analysed to confirm the assumption that the subjects' foreknowledge would be reflected in the overall performance. A doubly MANOVA was conducted on the session time and the number of clicks on a symbol in a session. Withholding or providing the following parts of knowledge were taken as between-subjects variables: Selector (2), Map (2), and Rotator (2). The only within-subject variable used in the analyses was Session (3). The analyses found main effects for Selector ($F(2,71) = 4.76$; $p. = 0.011$), Map ($F(2,71) = 10.97$; $p. < 0.001$) and Session ($F(4,69) = 32.09$; $p. < 0.001$). These findings show that the subjects' knowledge affected the way they interacted with the system. More specific analyses on the average session time combined with inspection of the means (Figure 2.12) revealed that less time was spent when the subjects received relevant Map ($F(1,72) = 4.88$; $p. = 0.030$) and Rotator ($F(1,72) = 6.04$; $p. = 0.016$) knowledge than when the subjects received irrelevant knowledge in the training. Furthermore, analyses

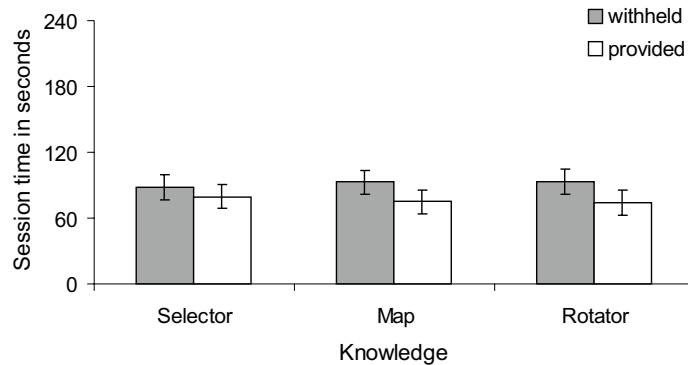


Figure 2.12: Time spent on average in a session with a 95% confidence interval.

on the averaged number of clicks combined with inspection of the means (Figure 2.14) revealed that the subjects made fewer clicks when they received relevant Selector ($F(1,72) = 6.52$; $p. = 0.013$), Map ($F(1,72) = 18.19$; $p. < 0.001$) and Rotator ($F(1,72) = 4.92$; $p. = 0.030$) knowledge, than when they received irrelevant knowledge. This means that the overall performance increased when the subjects could apply their knowledge.

The results of the prediction tests showed that the subjects' knowledge of the user interface increased after they performed the rotate task. Analysis of the performance over the sessions would give insight if this growth in knowledge coincided with more efficient task execution. Helmert Contrasts on the session time ($F(1,72) = 125.48$; $p. < 0.001$) and on the number of clicks ($F(1,72) = 76.19$; $p. < 0.001$) combined with inspection of the means (Figure 2.13 & 2.15) revealed that on average the subjects spent more time and made more clicks in the first session than in the later sessions. This effect seems to be a learning effect. However, the initial state of the system was different in the three sessions. Therefore, in theory, the effect found between the sessions can also be interpreted as an artefact caused by different initial states instead of solely being a learning effect.

2.3.3 Control on the lower-level layer

The clicks on the different symbols presented the user messages on the lowest-level layer in the structure of the fictitious user interface (Figure 2.6). Analyses of the number of clicks on the various symbols should show if they were differently affected by the subjects' knowledge of the different components, or in other words were affected by the usability of the components. Therefore, a doubly MANOVA was conducted on the number of clicks made on the following symbols: the bike, the aeroplane, the other symbols (fruit and musical instruments). Selector (2), Map (2), and Rotator (2) knowledge were again the between-subject variables and Session (3) the within-subject variable. The analysis found the following main effects: Selector ($F(3,70) = 4.36$; $p. = 0.007$), Map ($F(3,70) = 8.94$; $p. <$

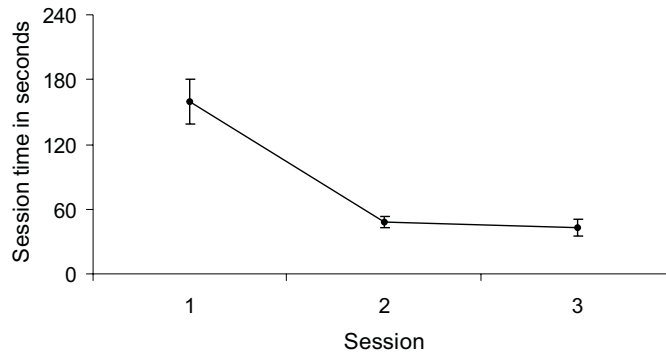


Figure 2.13: Time spent on average in each session with a 95% confidence interval.

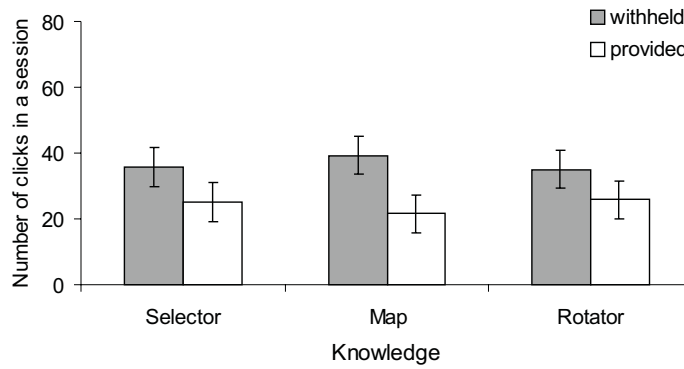


Figure 2.14: Average number of clicks made on a symbol with a 95% confidence interval.

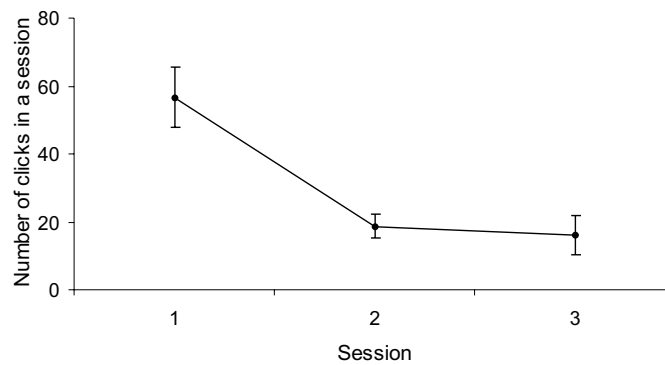


Figure 2.15: Average number of clicks made on a symbol in each session with a 95% confidence interval.

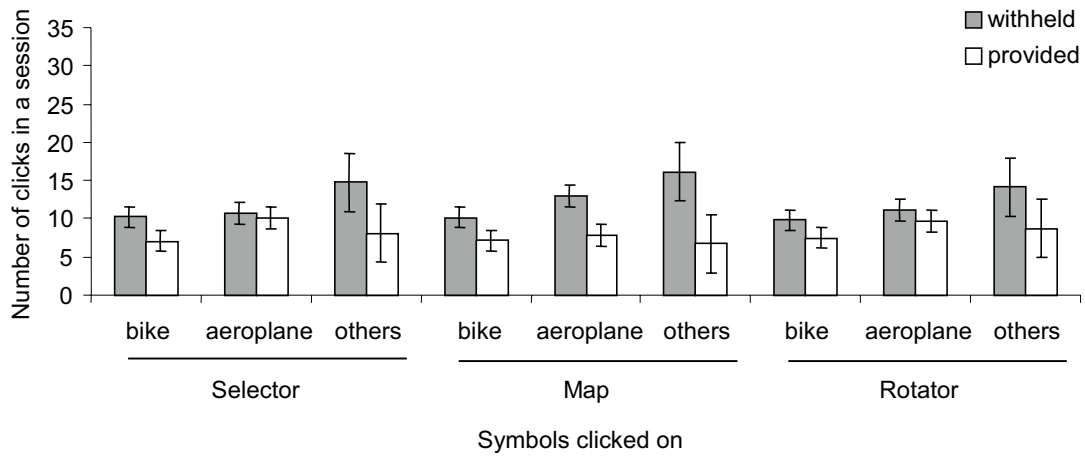


Figure 2.16: Average number of clicks in a session made on a bike, an aeroplane or another symbol with a 95% confidence interval.

0.001) and Session ($F(6,67) = 14.08$; $p. < 0.001$). The findings of this multivariate analysis are identical to those of the analysis of the overall performance measures. Therefore, more specific analyses were done on the number of clicks on each symbol averaged over the sessions combined with inspection of the means (Figure 2.16). The univariate analyses found no significant effect ($F(1,72) = 0.41$; $p. = 0.527$) for the Selector knowledge in the number of aeroplane clicks. However, it turned out that the subjects who received Selector knowledge in the training less often clicked on a bike ($F(1,72) = 11.17$; $p. = 0.001$) or on the other symbols ($F(1,72) = 6.06$; $p. = 0.016$) than the subjects who did not receive this knowledge. Furthermore, the subjects who received the Map knowledge less often clicked on the bike ($F(1,72) = 9.29$; $p. = 0.003$), on the aeroplane ($F(1,72) = 25.83$; $p. < 0.001$) or on the other ($F(1,72) = 12.08$; $p. = 0.001$) symbols, than the subjects that had not received this knowledge. Finally, the subjects who received the Rotator knowledge less often clicked on the bike ($F(1,72) = 5.97$; $p. = 0.017$) and on the other ($F(1,72) = 4.10$; $p. = 0.047$) symbols, than the subjects that did not receive this knowledge.

The results of the multivariate analysis on the number of clicks on the various symbols showed an effect for the sessions. This effect was also found in the univariate analyses of symbols clicked on. Helmert Contrasts on Session combined with inspection of the means (Figure 2.17) revealed that more clicks were made on the bike ($F(1,72) = 67.63$; $p. < 0.001$), the aeroplane ($F(1,72) = 48.90$; $p. < 0.001$) and the other ($F(1,72) = 60.50$; $p. < 0.001$) symbols in the first session than the later sessions. Again, it seems that an increase in knowledge reduced the number of clicks on all symbols.

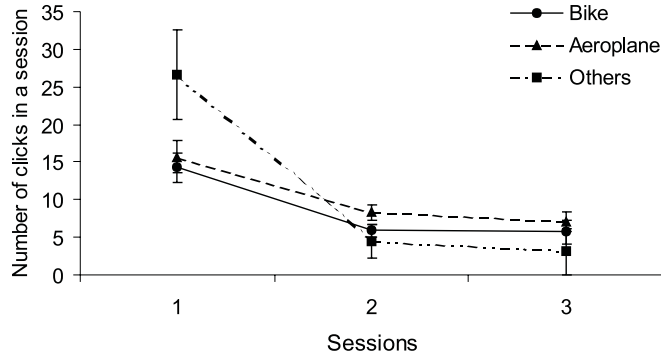


Figure 2.17: Number of clicks made on a bike, an aeroplane or other symbols in each session with a 95% confidence interval.

2.3.4 Control on the higher-level layer

Instead of the number of clicks on the aeroplane symbol, the translated message, ‘Rotate [objects]’, passed on by the Decoder to the Core of the Rotate component was analysed. The number of Rotate [target], Rotate [target-1] and Rotate [target-2] messages served as dependent variables in a doubly MANOVA. Once again, the between-subjects variables were Selector (2), Map (2), and Rotator (2) and the within-subjects variable was Session (3). The analysis found main effects for Map ($F(3,70) = 10.49$; $p. < 0.001$) and Session ($F(6,67) = 11.86$; $p. < 0.001$). These results agree with the previous observation that the knowledge related to the lower-level component did not affect the messages sent to the higher-level component, since no effect was found for the Selector knowledge ($F(3,70) = 0.35$; $p. = 0.79$). Again, the Map knowledge with its intermediate nature also affected the number of higher-level messages.

As was done with the analyses of lower-level messages, univariate analyses on the number of specific Rotate [object] messages averaged over the sessions combined with inspection of the means (Figure 2.18) were performed to study the effects on the specific Rotate [object] messages. The subjects who received the Map knowledge sent fewer Rotate [target] ($F(1,72) = 9.31$; $p. = 0.003$), Rotate [target-1] ($F(1,72) = 23.66$; $p. < 0.001$), and Rotate [target-2] ($F(1,72) = 10.98$; $p. = 0.001$) messages than those that did not receive this knowledge. Furthermore, the subjects who received the Rotator knowledge sent fewer ($F(1,72) = 4.10$; $p. = 0.047$) Rotate [target-1] messages than those who did not receive this knowledge. This latter effect indicates the usefulness of the analyses of the higher-level messages since no effect for the Rotator knowledge was found in the analyses of the number of click on the aeroplane symbol.

The extension of the subjects’ knowledge over the sessions was also found in the higher-level messages. Helmert Contrasts on Session combined with inspection of the means

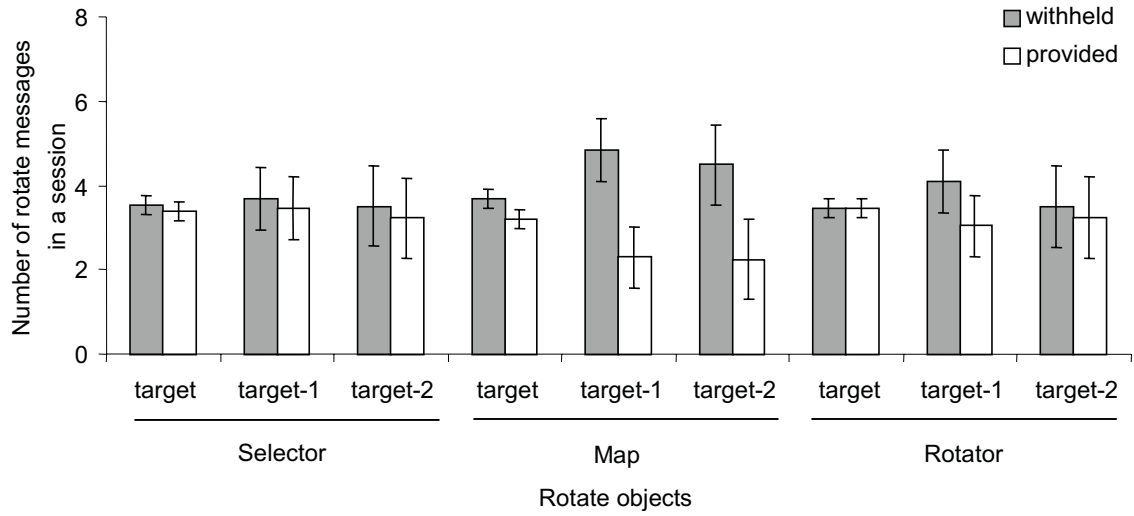


Figure 2.18: Number of rotate messages on average in a session made to rotate the target instrument, an instrument selected before the target (target-1), or the instrument selected even before that (target-2) with a 95% confidence interval.

(Figure 2.19) revealed that more Rotate [target] ($F(1,72) = 6.33$; $p. = 0.014$), Rotate [target-1] ($F(1,72) = 22.29$; $p. < 0.001$), and Rotate [target-2] ($F(1,72) = 5.23$; $p. < 0.001$) messages were sent in the first session than in the later ones, and that more Rotate [target-2] messages were sent in the second session than in the third session ($F(1,72) = 9.62$; $p. = 0.003$). Furthermore, in the number of Rotate [target] messages, a variation in the first session and the later sessions was found for a two-way interaction effect between Session and Map ($F(1,72) = 5.23$; $p. = 0.025$). Inspection of the means showed that between the first and later sessions the decline in the number of these messages was greater for the subjects who did not receive the Map knowledge compared to those that received it. The latter group of subjects already made almost no extra Rotate [target] messages after they rotated the required musical instrument.

2.4 Discussion

The results clearly show the expected effect on control by the knowledge that was obtained in the training by the subjects. The results of the analysis on their knowledge before and after the rotation tasks also make it likely that the knowledge gained by the experience of performing these tasks had a positive effect on the subjects' control. The analyses found no significant interaction effect for knowledge combinations on the control of the components. This means that in some cases the usability of components do not effect each other, which gives some justification for LPT's claim on the independence of components.

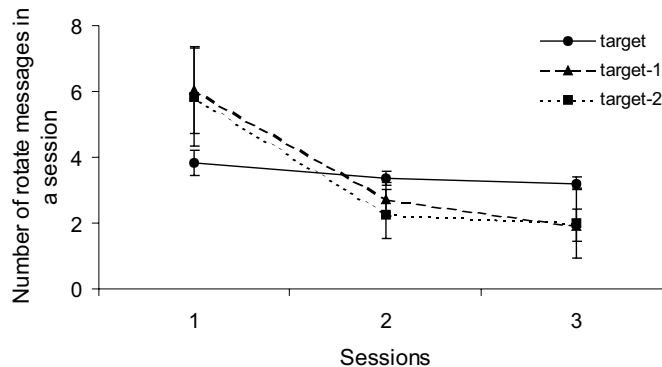


Figure 2.19: Number of rotate messages in each session made to rotate the target instrument, an instrument selected before the target (target-1), or the instrument selected even before that instrument (target-2) with a 95% confidence interval.

The expectations about the component-specific control did not seem completely correct.

A plain one-to-one relation between a component’s usability and a component-specific measure is a useful property for locating usability problems, especially in a usability test where only one version of a user interface is examined. The measure would directly reflect the usability of a component. However, the results show one-to-many relations between the subjects’ knowledge and the effect found in symbols clicked on (Figure 2.20). Variations in the number of clicks on a specific symbol could not solely be linked with the variation in a particular part of the knowledge, as was more or less expected (Figure 2.10). Still, some kind of pattern seems to emerge. The Selector knowledge did not affect the clicks on the aeroplane symbol. Since a click on the aeroplane symbol changed the state of the Rotate component, it seems that knowledge of lower-level component did not affect the messages sent to higher-level components. However, the opposite can not be said for the effect that the knowledge about the higher-level component had on the messages directed to the lower-level component. After all, an effect for the Rotation and the Map knowledge was found in the number of clicks on the bike symbol. A click on the bike symbol changed the state of the Select component, which related the bike symbol to this lower-level component. To explore the higher-level interaction, the subjects probably had to make more clicks on the bike to understand how this would affect the Rotate component indirectly.

At first, the Map knowledge seems to be more related to the higher-level component than the Rotator knowledge, because only a significant effect for the Map knowledge was found in the number of clicks on the aeroplane symbol. The subjects who could not draw the relation between the pieces of fruit and the musical instruments probably needed more exploration to understand how to rotate the target musical instrument, whereas the subjects without the Rotator knowledge only searched for the symbol that triggered a rotation. This explains the significant effects found in the number of clicks on a bike or other symbols and the lack

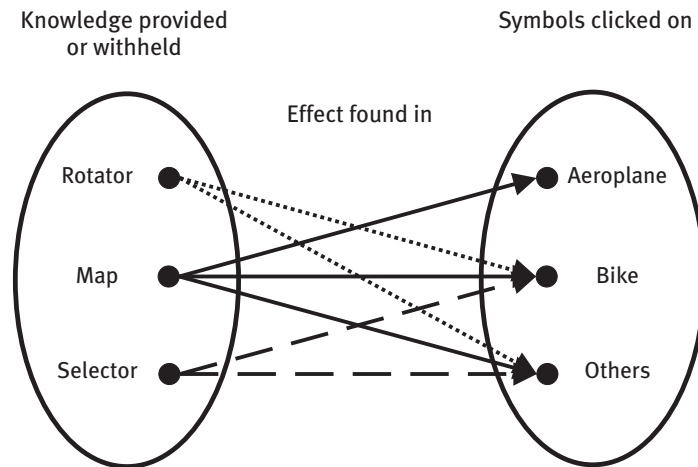


Figure 2.20: Significant main effects found for Rotator, Map and Selector knowledge in the three lower-level measures. A click on aeroplane symbol would trigger a rotation. A click on the bike symbol would result in the selection of another musical instrument. A click on the other symbols (musical instrument and fruit) caused no system state transition.

of a significant effect for the Rotator knowledge on the number of clicks on the aeroplane symbol. Clicking the aeroplane symbol would probably end the search almost directly, which could explain the limited systematic variation for the Rotator knowledge. The analysis of high-level rotate messages revealed the effect of the Rotator knowledge on the Rotate component. The higher-level analysis divided the variation in the number of aeroplane clicks across the three measures, probably reducing the unexplained part of the variation and allowing the variation for Rotator knowledge to be more noticeable.

The Map knowledge cannot be solely allocated to the Rotate or to the Select component. Instead, it demarcated the border between these two components, in which a piece of fruit transformed from a symbol that can be changed into a symbol that refers to a musical instrument. Because this knowledge had a clear impact on the control, a place in the compositional structure of the user interface should be considered.

2.5 Conclusions

One of the main conclusions that can be drawn from this explorative experiment is that the number of messages a component receives is a potential objective component-specific control measure. The results show that the subjects' knowledge of a component directly affected the number of user messages the component received. The measure reflects the users' effort to control their perception of a component. However, the experiment also showed that this is not a plain one-to-one measure, since an effect for the subjects' knowledge of the higher-level component was found in the number of lower-level messages. Still,

this measure is potentially useful when it can signal usability differences earlier than overall performance measures do.

Another reason for conducting the experiment was to see whether the Decoder-Core-Coders arrangement provides an adequate way of describing the interaction and the user interface architecture. It seems too extensive when only the number of user messages is studied. Instead of specifying all elements of a component and their mutual message exchange, the message exchange between the components would be sufficient to understand the compositional structure. Furthermore, the compositional structure should be presented with a method that is more common in the design field, so more designers could make use of the achieved knowledge. The unified modeling language (UML) (Booch, Rumbaugh, & Jacobson, 1999) might be a good candidate as already suggested by Paternò (2000).

No interaction effect was found between the different knowledge parts, which supports the claim of LPT that components do not affect each other's usability. However, it should not be concluded that this never can be the case. The experiment was not set up to push the components' dependencies to the limit by introducing very plausible combining factors. Further research is needed to understand the independence of a component.

The research described in the following chapters aims to confirm to what extent the control measure is component specific. In an attempt to generalise this measure into a usability measure, other usability factors besides foreknowledge are studied. Consequently, a more realistic user interface is considered to ensure the validity of the measure. This also makes it possible to search for other component-specific measures next to this objective one, such as perceived ease of use or the satisfaction.

Chapter 3

Usability testing in a component-based environment

3.1 Introduction

After implementing a prototype, software engineers need a usability test to answer two questions: are changes required in the user interface, and if so, what exactly should be changed? The first question can be answered by the currently available usability measures. These are overall measures since they measure the usability of the entire user interface. The second question is a much harder one. Evaluators can estimate the task duration or the number of keystrokes, but have to make an educated guess about the troubling part of the user interface. Several authors (Taylor, 1988a, 1988b; Haakma, 1998; Hilbert & Redmiles, 2000) have proposed the Layered Protocol Theory (LPT) as a candidate for analysing user behaviour to assess the usability of individual parts of a user interface. Usability measuring per protocol is expected to be even more effective in finding usability difference between two protocol variants than overall measures (Haakma, 1998). Measuring usability at various interaction layers is feasible because LPT treats protocols as user interfaces in their own right. To reduce the amount of work and to increase the reliability of usability measurements, Haakma suggests employing automatic procedures to determine the performance-oriented aspects of usability on the basis of usage logs.

3.1.1 Objectives evaluation method

The main objective of the usability evaluation method described in this chapter is to create a testing framework in which user interface components can be evaluated for their usability. The testing framework can be used in the following two situations: in cases where multiple versions of one component are compared, or in cases where entire user interfaces are evaluated. In the first situation, the benefit of the evaluation method lies in its potential power in finding usability differences. In the second situation, the benefit lies in its ability

to evaluate the usability of the components individually, something overall measures can not offer in this situation. Another aim of the method is to provide a reliable usability evaluation. Therefore, the method includes a description of how to analyse the components systematically. For its validity, the method appeals to its underlying theoretical principles of LPT and Perceptual Control Theory (PCT).

3.1.2 Overview of the evaluation method

A testing framework is proposed in this chapter for component-based usability testing. As a starting point, the framework takes the elementary components of user interfaces on which behavioural-based evaluation is possible, and defines their places in the control hierarchy based on the required mediation of other components to interact with users.

The testing framework supports two testing paradigms, the single version and the 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 usability of the user interface. In the multiple versions testing paradigm, different versions of components are compared with each other. The question in this case is which version has the highest usability.

The testing framework includes component-specific perceived ease-of-use, satisfaction, and performance measures. The component-specific perceived ease-of-use and satisfaction measures are obtained through a standard questionnaire. These component-specific measures are expected to be more powerful because the specific questions can assist users in the recall of their control experience of individual components.

The number of messages a component receives is proposed as the component-specific performance measure, which reflects the users' effort to control their perception of a component. In the multiple versions testing paradigm, the power of this component-specific measure comes from the reduction in statistical variance by limiting its focus to one control loop, and consequently, lock out the variance caused by the users' effort to control other components. For the single version testing paradigm, the number of messages a component receives is set against the performance of an ideal user and is corrected for control effects of lower and higher-level components. The measure allows evaluators to order the components according to their usability improvement potential. However, when it comes to applying this measure, it is presumed that components only receive messages that were sent with the intent to control the components. The Standardised Reception Coefficient is introduced to help evaluators check whether messages were sent unintentionally as a side effect of lower-level control loops.

3.1.3 Overview chapter

The following section gives a formalisation of LPT for a component-based environment. The formalisation provides a frame in which the evaluation method can operate. The ar-

chitectural elements of a user interface are presented, and the effects control loops may have upon each other are further examined. After this, the evaluation method is described in more detail. The description starts with the test procedure and moves on to the component-specific measures. First, the objective performance measures and then subjective measures are discussed. The objective measures for the two paradigms are discussed separately. The chapter ends with comparing the method with other evaluation methods.

3.2 Formalising LPT

3.2.1 Architecture

The following three sections introduce the concepts: interaction components, layers, and routers. With these concepts user interface architectures can be described.

Interaction component

Interaction components define the elementary units of user interfaces, on which behaviour-based evaluation is possible. An *interaction component* is a unit within a device that directly or indirectly receives signals from the user. These signals enable the user to change the physical state of the interaction component. Furthermore, the user must be able to perceive or to infer the state of the interaction component. Therefore, an interaction component should provide I-feedback. Without the possibility of perceiving the state, users cannot separate the component from the whole system and are not able to control it. Without the ability to control their perception, users' behaviour is pointless. E-feedback, although often essential for easy control, is not required for behaviour-based evaluation.

The idea of a physical state agrees with the notion that interaction components are the instantiation of functional definitions (Holyer, 1992; Kobryn, 2000). An interaction component is not a concept or an abstract description in someone's mind of how a system operates; it is a physical entity in the world. It may be very prominent like mechanical parts in a device, or it may be present in the background like an electric charge inside a chip. Furthermore, its state should be controllable. A *minute* label of an alarm clock is not an interaction component on its own because users cannot change it (Figure 3.1). A behaviour-based assessment of the quality of this label can only be made as part of an interaction component responsible for the minute digits, whose state users can control. From this point onwards, the graphical representation of Unified Modified Languages (UML) (Booch et al., 1999) for a component is used to represent interaction components (Figure 3.2).

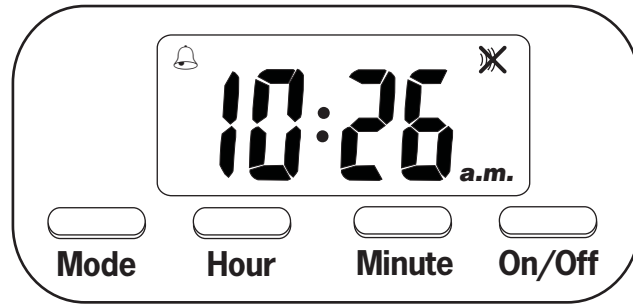


Figure 3.1: The front of an alarm clock, in which the alarm time is visible and the alarm is deactivated.

Layers

The points where input and output of different interaction components are connected demarcate the border between layers. An interaction component operates on a higher-level layer than another interaction component, when the higher-level interaction component receives its user's messages from the other interaction component. With the definition of interaction components and layers it is possible to describe the architecture of a regular alarm clock (Figure 3.1) to illustrate these concepts. With this alarm clock, users can set the alarm time by setting the clock in the right mode, changing the hours and minutes digits, and finally activating the alarm. Figure 3.2 shows the interaction components involved. The Hour and Minute interaction components are located on the lowest-level layer, where they manage the state of the hour and minute digits. The Mode interaction component is placed on the middle-level layer. The component is responsible for the mode of the alarm clock, and consequently whether the current or the alarm time is visible or set. To indicate that the alarm time is displayed, a small icon of a bell is shown in the top-left corner of the display (Figure 3.1). The Alarm Time and Current Time interaction components, which keep the corresponding times, make up the top-level layer in the architecture. The Alarm Time interaction component shows a small icon in the top-right corner of the display (Figure 3.1) to indicate whether the alarm is activated or not.

Router

Another element in the architecture of user interfaces is the Router. Routers are binding elements that direct the communication flow between interaction components, and do not have to have an own state. The connection between lower and higher-levels interaction components can either be direct, as shown in Figure 3.3A, or indirect, as shown in Figure 3.3B. The in-between component in Figure 3.3B is a Router. This component's only function may be merging of the messages of the two lower-level interaction components into a message for the high-level interaction component. Routers can transform (or redirect)

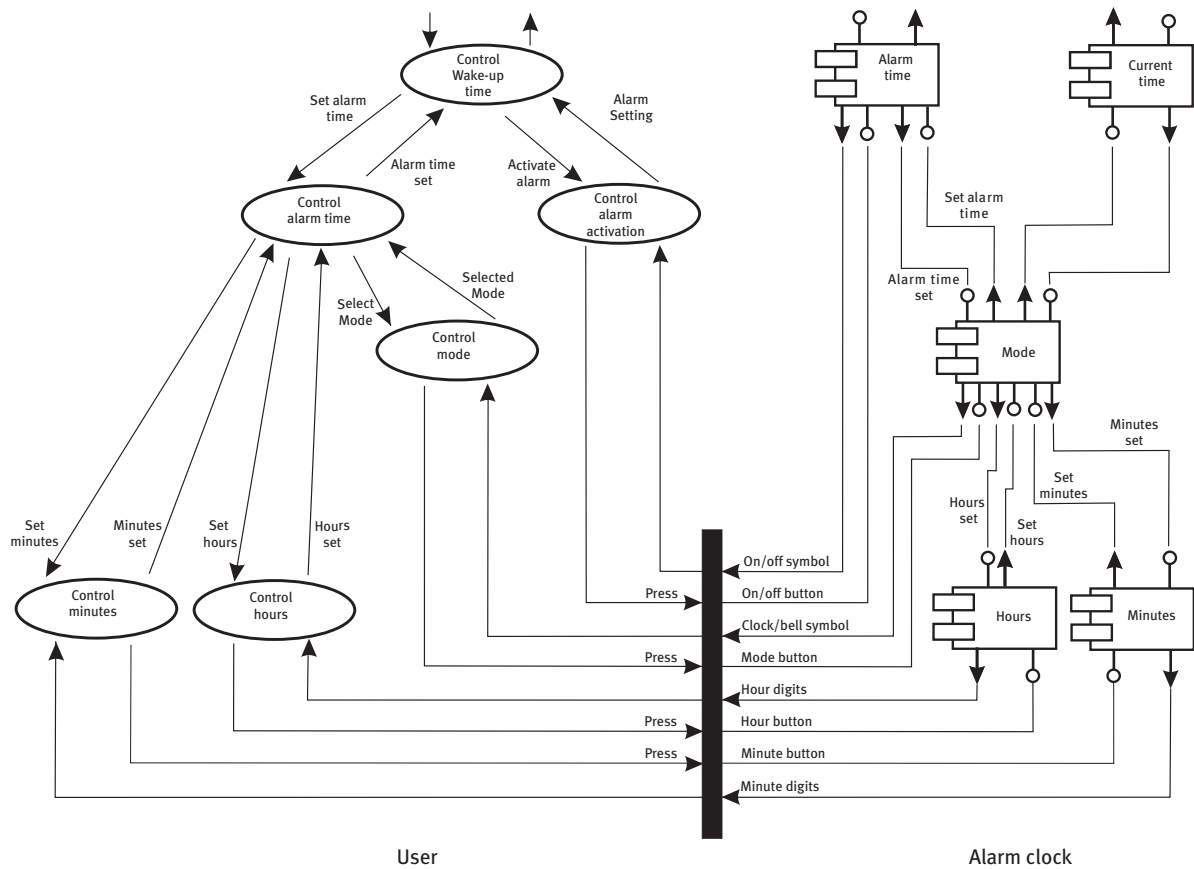


Figure 3.2: Layered interaction structure between a user and an alarm clock, when setting the alarm time. The Expectation feedback is left out.

output messages from different interaction components into input messages towards other interaction components. Although they are not required to have their own physical state, they may copy and store the state of other components to transform input messages or to direct output messages. A router can be both a multiplexer and a demultiplexer (Taylor & Waugh, 2000).

Users are unaware where the merger or splitting of messages exactly takes place in the architecture. Consider the alarm clock once again with its Minute and Mode interaction component. Whether a press on the minute button should result in a message to set the current time or the alarm time one minute later will be determined when the information from these interaction components comes together. For all the users know, they come together inside the higher-level interaction component itself (Figure 3.3A) or inside one of the lower-level interaction components (Figures 3.3C and 3.3D) that receives the output of the other lower-level interaction component. Note that the architecture in Figure 3.3C and 3.3D even has an extra layer. Only the software engineer knows which component is really responsible for the merger; the architecture does not enforce a specific location.

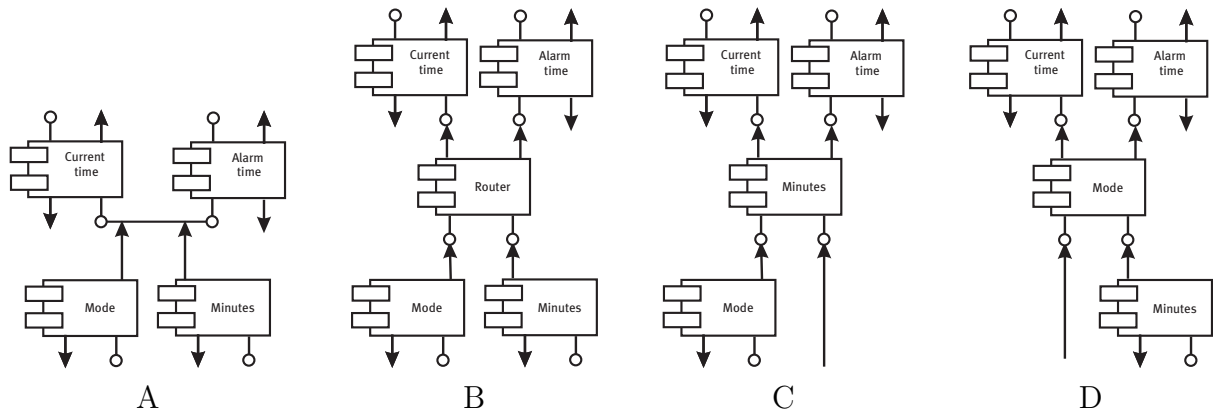


Figure 3.3: Various ways in which the higher-level interaction components can be connected with lower-level interaction components.

Although users may not control the router, the coupling knowledge helps users to control their perception of a higher-level interaction component. When deciding whether or not to press the minute button of the alarm clock, it helps to know that the alarm time currently displayed will change and not the current time. Therefore, by specifying these couplings the factors become more clear that influence the user's ability to control their perception.

3.2.2 Control effects

There are basically two ways in which the user interface architecture itself can cause the number of messages received by an interaction component to be affected by another interaction component. An interaction component can affect lower and higher-level interaction components by requesting more low-level messages or sending more messages upwards than required. The first is defined as inefficiency, the second as ineffectiveness.

Efficiency

An efficiency improvement would only have an effect on the interaction component itself and its lower-level interaction components because the same number of messages would be sent to higher-level interaction components. Figure 3.4 shows this situation. Interaction component X has an efficiency problem. As a result, more low-level messages are requested. To make this higher-level effect more concrete, take for example the situation where users want to hear the alarm of the alarm clock; but, when they set the alarm time to go off the next minute, they forget to activate the alarm. Once the new minute comes and the alarm clock remains silent, users may realise that they also have to activate the alarm. They set the alarm clock for the following minute and activate the alarm this time. Because of the inefficiency problems with controlling the higher-level Alarm Time interaction component,

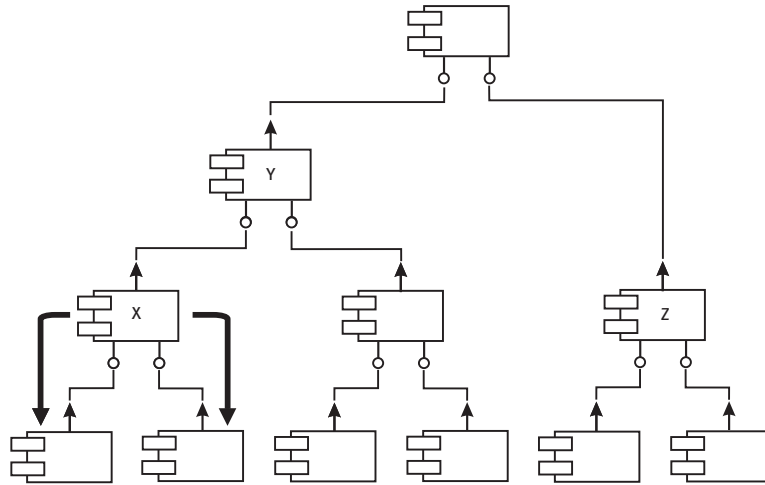


Figure 3.4: The lower-level interaction components of interaction component X will receive fewer messages after an efficiency improvement of interaction component X.

the users also had to interact with the lower-level Minute interaction component more times.

Effectiveness

A problem in an interaction component may also unintentionally increase the number of messages sent upwards. Variation in the number of messages unintentionally sent upwards is defined as variation in effectiveness. Take for example the case where users would misunderstand the bell icon in the alarm clock, and set the current and alarm time in exactly the opposite way to that intended. Once they recognise the error, they have to reset both the current and alarm times. The Mode interaction component, which manages the mode state, operates ineffectively because more messages need to be sent upwards to both higher-level interaction components than would be required if the users had not mixed up the modes of the alarm clock. Effectiveness problems could trigger more messages to undo the effect of the initial message sent upwards. These recovery messages might even come from other lower-level interaction components than the interaction component that created the problem in the first place. Figure 3.5 illustrates how a usability problem of the interaction component X can spread through the system. Interaction components that have a common higher-level interaction with interaction component X can be faced with the ineffectiveness of interaction component X.

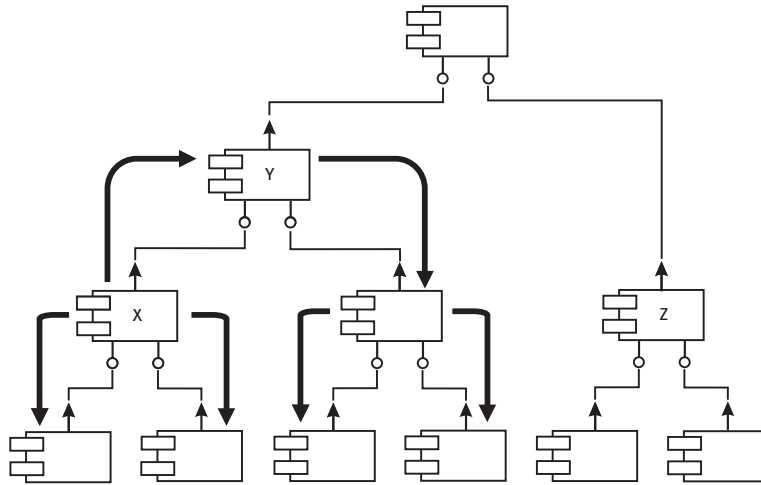


Figure 3.5: Higher-level interaction component (Y) is affected by ineffectiveness of interaction component X, whereas interaction component Z is not. The dark arrows show which interaction components will be affected.

3.3 Evaluation method

3.3.1 Test procedure

The test procedure of the evaluation method roughly corresponds to the normal procedure of a usability test. Subjects are observed while they perform a task with a system. In the case of multiple versions testing paradigm, the subjects perform the same task but with different versions of a system. The versions are created by applying different versions of the interaction component in the system and keeping the rest of the system the same. A task is finished once subjects attain a specific goal that would require them to alter the state of the interaction component under investigation. In advance, subjects are instructed to act as quickly as possible to accomplish the given goal. Before the test, a threshold time should be determined after which the experimenter helps the subjects, otherwise they might end up trying to solve a task endlessly. The threshold time might be based on the performance of subjects in a pilot study. A threshold time can be the average task time plus three times the standard deviation. This threshold is often used in statistical analyses to find outliers. As subjects perform the task, messages sent to the various interaction components are recorded in a log file. Once they reach the goal, the recording stops. After the subjects perform the task, they fill out a questionnaire about the perceived ease-of-use and the satisfaction.

The message exchange between the higher-level interaction components can also be obtained afterwards, if the low-level events only are recorded. In that case, a conversion process should construct the higher-level messages. An example can be found in the work

of Docampo Rama (2001). She applied a conversion tool to transform a log file that consisted of only the button presses on a television remote-control prototype to a log file with high-level description of the user's executed actions. As the conversion tool read the low-level action log, it simulated parts of the internal message exchange of the prototype and logged this into a new log file.

3.3.2 Objective performance measure in the multiple versions testing paradigm

Once the task is completed, the normal overall measures, such as task duration and total number of keystrokes, can be calculated from the log file. In addition, the number of messages received by an interaction component can be calculated, which is put forwards as a component-specific performance measure within this paradigm. The explorative study described in chapter 2 indicated that the interaction component version that received the fewest messages is the most usable one. The number of messages indicates the users' effort to control their perception of the interaction component, provided that the subjects attained only one goal. Therefore, once they reached the goal, the measuring is stopped. Otherwise, an increase in the number of messages can also indicate the users' ability to control their perception better as they achieve more goals (e. g. Neerinx & Greef, 1998).

Taking the analysis of the total number of keystrokes as a point of reference, the component-specific performance measure is expected to increase the statistical power of the evaluation method. Statistical power in this case is defined as the chance of finding a difference provided there is a difference. The power of the measure is increased because the variance in the data is reduced by looking only at the number of messages received by an interaction component instead of the total number of keystrokes received by the whole user interface. The additional variance, created as users try to control other interaction components, is left out. This variance reduction can be apparent in the analysis of lower-level interaction components, but this can apply to higher-level interaction components as well. A low-level message does not always lead to a high-level message. Therefore, the effect of replacing a high-level interaction component can be more obvious in the number of high-level messages than in the number of keystrokes. The effect size might be too small to be discriminated in an analysis of variance on the number of low-level messages.

Example

Consider a drawing application. In this application, users can change the typeface of texts they have put in their drawing. After users select a text, they can use a Right Mouse Button Menu (RMBM) to activate a property window from there. In the property window the users can change the typeface. When software engineers are not sure how to present the typeface in the property window, they can make two versions, and test them: one that only states the name of the typeface, and another one that also gives an example text

of the typeface. The most usable version has the smallest number of messages received. This variation is easier to notice than the variation in the total number of mouse-clicks received by the whole application. The latter one fluctuates more because it is affected by all other control problems in the user interface. Therefore, monitoring the number of messages received by the property window is more effective in choosing the best version than monitoring the low-level messages.

Limitations

The test assumed that the users have to spend the same amount of effort each time they send a message on the level of the interaction component. When high-level interaction components are tested, this assumption is reasonable between the two versions, because the low-level interaction components are the same. However, when the lowest-level interaction components are evaluated, more attention should be given to this point. Take for example the evaluation of the *sam* text editor by Thomas (1998). 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 application reported in the literature recorded action on a higher-level layer (e.g. sending command lines). He concluded that different things were measured, and therefore that they were difficult to compare.

The total number of keystrokes could be more powerful than the component-specific measure when the usability variation of one interaction component influences the number of messages received by another interaction component. This can be caused by three factors: the user, the environment and the user interface architecture. Because of poor usability, users can start misusing other parts of a device while keeping the number of messages sent to the targeted interaction component the same. The cause is twofold, the E-feedback of neither two interaction components triggers the proper user's intention and actions. This problem may be observed in the total number of keystrokes, while it is not seen in the input of the interaction component itself.

Besides the user, the environment is a factor too. Users can use a device to change their perception of the environment. The environment can for example be another person. Users may try to send a text message to a person with their mobile telephone. Imagine that when this person tries to understand the text message he or she runs into the problem that the message is not interpretable because of many typing errors. Instead of sending back a text message asking to repeat the message with the risk of again receiving a message with a lot of typing errors, the receiver calls the user of the mobile telephone and asks him or her to give the message verbally. The number of calls the user received is now related to the usability of the interaction component responsible for making a character. The statistical power of the keystrokes would in this example be greater than that of the number of messages received by the troublesome interaction component.

Finally, an analysis of the number of keystrokes can sometimes better pick up ineffec-

tiveness problems than of the number of messages received by an interaction component, because multiple interaction components can be involved as the problem spreads through the system.

3.3.3 Objective performance measure in the single version testing paradigm

In the single version testing paradigm, the performance of a (fictional) ideal user is compared with that of subjects. On the assumption that lower-level layers have no effectiveness problems and messages sent by an ideal user are also sent by real users, it is possible to estimate the performance of an interaction component in the single version testing paradigm. The estimation only looks at performance of the interaction component as if the higher-level interaction components operated optimally. This way the inefficiency of higher-level interaction components is compensated for.

Valuable information for evaluators in the single version testing paradigm is related to which interaction components should be changed to create the largest usability improvement of the whole system, i. e. impact assessment and effort allocation (Hilbert & Redmiles, 2001). The impact an interaction component has on the overall performance can be estimated by assigning an effort value to each message received. These effort values are based on the effort value of the lowest-level messages that are linked to higher-level messages. At the lowest-level layer, weight factors of an arbitrary unit are assigned to the messages, which present the user effort value to send a single message. This can be a number for each keystroke or for the time users need to make an elementary message. The users' effort, to make these elementary actions, is regarded as similar throughout task execution. By expressing all elementary messages in time, instead of keystrokes, analyses can even compare messages from different input devices.

In the next section, an example is worked out to explain the performance evaluation in detail. The referred formulas and predicates can be found in the appendix A on page 181.

Example

Description of the application and the task Imagine the drawing application again. A circle and polygon are drawn; both the circle and polygon are white and have a black outline. Now, the task is to change the circle's colour into red and remove its black outline. Table 3.1 shows the actions and the application response to perform this task optimal. If users perform the task optimally, they first select the circle by clicking on it, then they open the RMBM and choose the option Properties. The application comes up with a property window, where the users select the red box in the Fill tab sheet (Figure 3.6). They also check the No Line check box on the Outline tab sheet. Finally, the users click on the Ok button of the property window and the circle changes accordingly. Figure 3.7 presents the

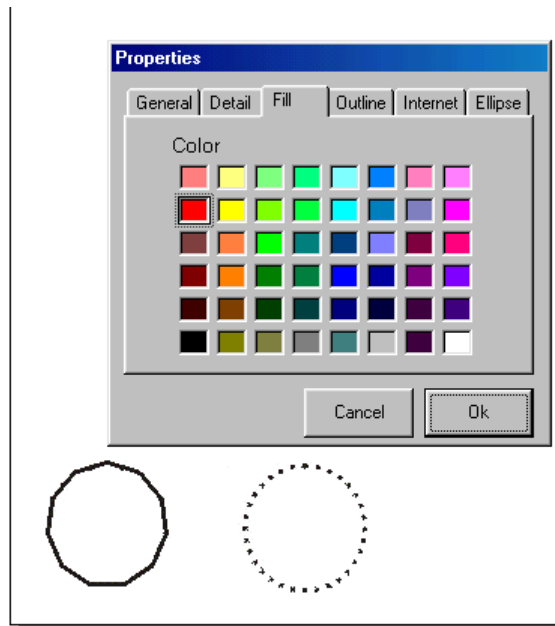


Figure 3.6: Property window for setting the fill colour of the selected circle to red. The selection is made visible by a dashed outline of the object.

message exchange between the relevant interaction components of the drawing application when the task is conducted in the optimal way.

In this example, (imaginary) recordings are also made of the behaviour of an (imaginary) real user who has three problems: first, with selecting an object; second, with setting the right properties; and third, with distinguishing a circle from a polygon. Table 3.2 shows the real user's actions. The real user selects objects with a selection window instead of clicking on the objects themselves. Furthermore, the real user takes the polygon for a circle, and consequently makes two selections. After the circle is selected, the real user changes the borderline's colour into red by mistake, instead of the circle area. Figure 3.8 shows the message exchange between the same interaction components as before, for the real user.

Analysis of the example In the circle example of the drawing application, a value 1 is assigned to all Click messages (Figure 3.7 and 3.8). After effort values are assigned to the lowest-level messages, the effort values are calculated for the messages sent upwards. The effort value of a message sent upwards is, in principle, the sum of the effort values of the messages received between this and the previous message sent upwards (Appendix, Predicate A.11). Figure 3.7 gives the effort values for the messages sent upwards in the case of the ideal user. Take for example the Call message sent upwards by the RBMBM at event 4. Two Click messages, each having an effort value 1, were received before this

Table 3.1: Actions and application responses of setting the fill colour of the circle to red and removing its borderline in the case of optimal task execution.

User action	Application response
click left on circle	circle marked (dashed outline)
click right	Right Mouse Button Menu open
click right on Properties option	menu closed, property window open
click left on Fill tab	Fill tab sheet visible
click left on colour red	red box marked
click left on Outline tab	outline tab sheet visible
click left on No Line check box	No Line check box checked off
click left on Ok button	circle red without borderline

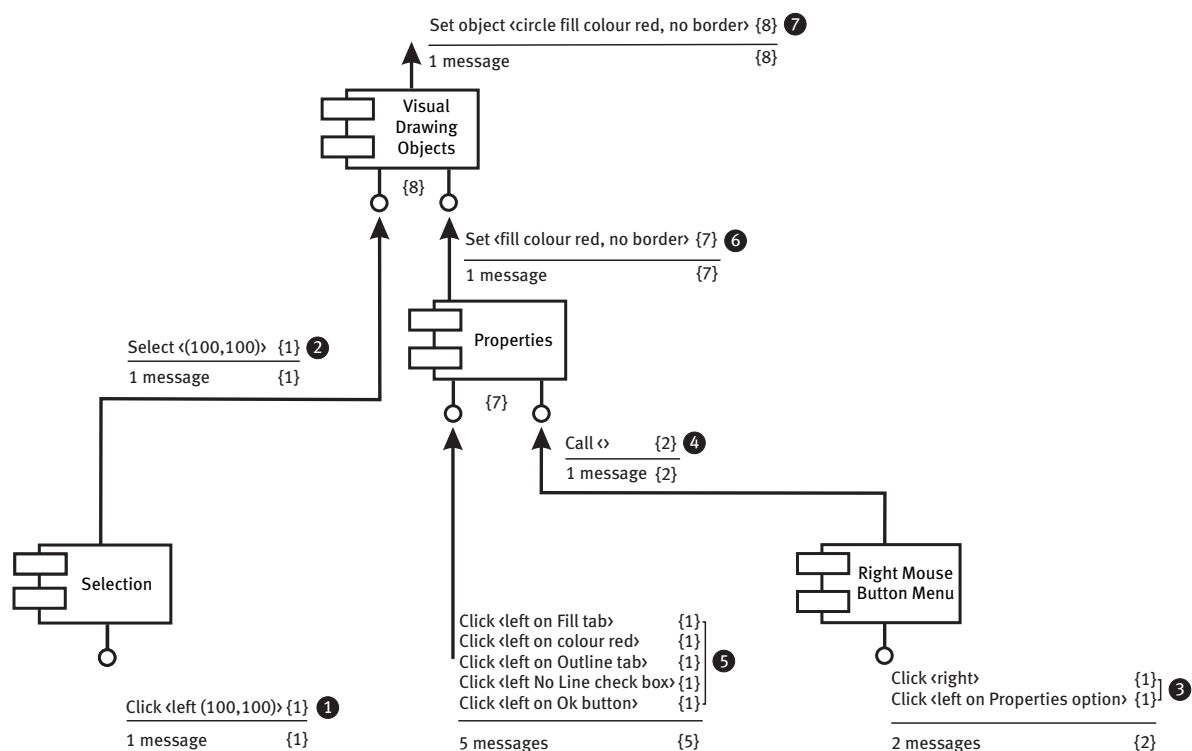


Figure 3.7: Part of drawing application's architecture, with only relevant interaction components and connections for the task of setting the properties in the case of optimal task execution. Effort values are given within brackets. The numbers in the black circles give the event sequence of the task execution.

Table 3.2: Actions and application responses of setting the fill colour of the circle to red and removing its borderline in the case of an observed task execution.

User action	Application response
click left below polygon with button kept down	corner of selection window visible
button released on spot right above the polygon	polygon marked (dashed outline)
click left below circle button kept down	corner of selection window visible
button released on spot right above the circle	circle marked (dashed outline)
click right	Right Mouse Button Menu open
click right on Properties option	menu closed, property window open
click left on Outline tab	outline tab sheet visible
click left on colour red	red box marked
click left on Ok button	borderline circle red
click right	Right Mouse Button Menu open
click right on Properties option	menu closed, property window open
click left on Fill tab	Fill tab sheet visible
click left on colour red	red box marked
click left on Outline tab	outline tab sheet visible
click left on No Line check box	No Line check box checked off
click left on Ok button	circle red without borderline

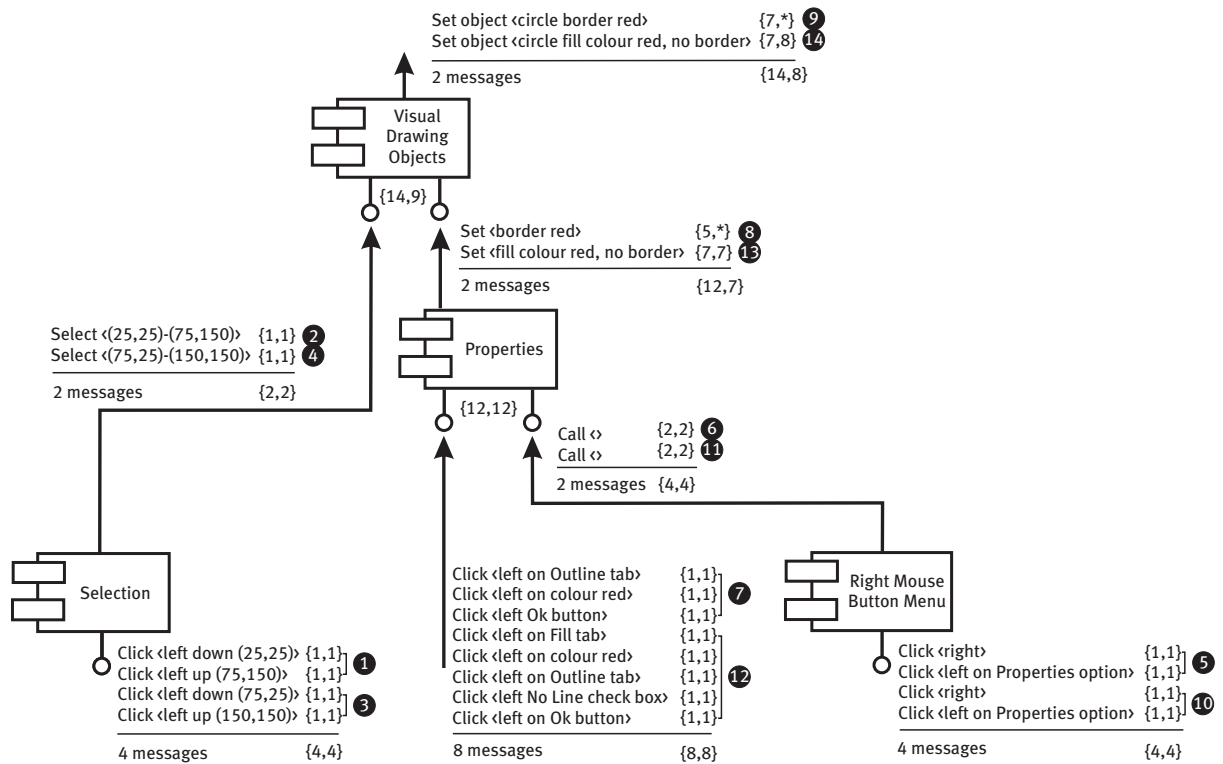


Figure 3.8: Part of drawing application’s architecture, with only relevant interaction components and connections for the task of setting the properties in the case of an observed task execution (real user). Effort values are given within brackets; the second value within the brackets is the effort value according the conservative method. The ‘*’ means that the message is ignored in the conservative method, because the message’s effect is different from that of messages with the same type sent in the case of optimal task execution. The numbers in the black circles give the event sequence of the task execution.

Table 3.3: Number of messages sent upwards in both optimal and observed task executions, followed by total effort values, user effort, and finally the extra user effort.

	Sent upwards		Total effort		User effort	Extra
	Optimal	Observed	Optimal	Observed		
Selection	1	2	1	4	2	1
RMBM	1	2	2	4	2	0
Properties	1	2	7	12	6	-1
VBO	1	2	8	14	7	-1

message was sent (see event 3); therefore, an effort value of 2 is assigned to this message. The calculation starts at the lowest-level layer and works its way upwards. This means that first the effort value for the Selection and RMBM interaction components is calculated, followed by the effort value for the messages sent upwards by the Properties and finally by the Visual Drawing Objects (VDO) interaction component.

If messages are sent upwards through actual task execution (the real user), the inefficiency of lower-levels is removed from the effort value. The effort values of the messages that are also sent upwards in optimal task performance receive a similar effort value as if these messages are sent upwards in optimal task performance (Predicate A.12). If another type of message or a message with another effect is sent upwards compared to that in optimal task performance, it will receive the sum of the effort values of messages received. However, if this message is of the same type as messages sent upwards in optimal task performance, it may not exceed the maximum optimal effort value of messages of this type. For example, the Select messages, sent upwards by the Selector in Figure 3.8 at event 2 and 4, have an effort value of 1 because the ideal user made a Select message that has the same effect with only an effort value of 1.

The total effort for the control of an interaction component is the sum of the effort values of the messages received (Equation A.13 and A.14) (third and fourth column of Table 3.3), which, in the case of real users, is later on corrected for the inefficiencies of a higher-level (Equation A.15). Without correction, the total effort still may include inefficient high-level demands, which should not be charged to the interaction component. Therefore, the analysis only looks at the number of messages sent upwards that are required to fulfil the optimal request of higher-level layers. This corrected value is called the user effort and is also given in Table 3.3. It is calculated by multiplying the number of messages the ideal user sent upwards (first column) with the total effort in the case of the real user (fourth column) divided by the number of messages the real user sent upwards (second column). This correction assumes that the same number of messages have to be received to send a message upwards again, which is the case when the state of the interaction component is reset after a message is sent upwards. When an interaction component state is not reset

after sending a message, it may in some cases still be possible to determine the optimal user effort. At the moment the message is sent upwards, the interaction components may determine how much effort an ideal user would make to create this message and record this in the log. Additionally, it can take into account the interaction component state when the previous message was sent upwards, or lower-level messages received that cancel each other out.

The extra user effort is a measure of the effort difference when a real user or an ideal user controls an interaction component. The extra user effort is the result of the subtraction of the total effort made by the ideal user from the user effort made by the real user (Equation A.16). Table 3.3 shows the extra user effort per interaction component. The Selection interaction component is charged with 1 extra keystroke. Although the real user made an inefficient selection two times, only the average extra user effort per selection is taken into account because the repetition is attributed to inefficiency of higher-level layers. The RMBM interaction component, with its optimal performance, is subsequently charged with no extra user effort. The extra user effort of the other two interaction components is dubious because of the ineffectiveness of the Properties interaction component. Setting the borderline's colour to red, instead of the area (event 8) causes both the Properties and VDO interaction components to end up with a negative result. Therefore, interpretation requires the assumption that lower-level layers perform effectively, i. e. messages are only sent upwards to fulfil the demands of higher-level layers and not as a result of a usability problem of the interaction component itself.

Finally, ordering the interaction components by their extra user effort creates the priority list of the most effective improvements.

A more conservative analysis of the example The effort value of messages sent upwards can also be calculated in a more conservative manner, which is less affected by the ineffectiveness of lower-level layers. The idea with the conservative view is that only the messages sent upwards that are regarded as correct input can be attributed to the effort value of the higher-level interaction component. The lower-level interaction component is responsible for the creation of ineffective messages, and the impact these messages have on other higher-level interaction components should be limited as much as possible. Therefore, a message sent upwards is removed when, in optimal task execution, all the messages of this type have another effects on a higher-level interaction component than this message has (Figure A.1).

Figure 3.8 also gives the conservative effort value of the real user in the circle example of the drawing application (second value within the brackets). The Set message with the content *<border red>* sent by the Properties interaction component is ignored (see event 8). This Set message has a different effect from the only Set message *<fill colour red, no border>* sent by the ideal user. Consequently, the Set Object message *<circle border colour red>* sent upward by the VDO is also ignored (event 9). Furthermore, the effort value of the two selection messages received by the VDO is attributed to the Set Object message

Table 3.4: Number of messages sent upwards in both optimal and observed task executions, followed by total effort values, user effort, and finally the extra user effort according to the conservative method.

	Sent upwards		Total effort		User effort	Extra
	Optimal	Observed	Optimal	Observed		
Selection	1	2	1	4	2	1
RMBM	1	2	2	4	2	0
Properties	1	1	7	12	12	5
VDO	1	1	8	9	9	1

<circle fill red, no border>, which is later sent upwards at event 14. Although the sum of the effort values of the four messages received is nine, a value of only eight is assigned—the optimal effort value. The ignored messages are not counted either when it comes to the observed number of messages sent upwards (Table 3.4, second column), which has its impact on the user effort. The last column of Table 3.4 shows the extra user effort calculated according to the conservative method. The results of the Selection and RMBM interaction component are the same as before. Quite the opposite can be said of the results of the other two interaction components. The Properties interaction component is charged with five extra keystrokes; the five keystrokes an ideal user would need to set the border colour of the circle to red. The VDO interaction component is charged with only one extra keystroke; the one keystroke, an ideal user would need to select the polygon.

The conservative view has the advantage of being more robust or, put into other words, less sensitive to lower-level ineffectiveness. However, if the lower-level interaction component sends more *correct* messages upward than required, it still can disturb the indication of the user effort. Furthermore, the conservative view has the drawback of ignoring messages sent upward with other effects, which users may send to test how the higher-level interaction component works. If this effort should be attributed, it should certainly be attributed to the higher-level interaction component. Still, the conservative view regards these messages as the result of a problem on a lower-level layer.

Limitations

The performance measure in the single version testing paradigm has at least the same limitation as the performance measure has in the multiple versions testing paradigm. Factors like the user and the environment can make relations between the interaction components that can disturb the usability analysis. As illustrated in the worked out example, an ineffectively operating interaction component can also make it difficult to interpret the results for the interaction component, but also for the interaction component one layer higher.

However, instead of performing the complete calculation of the extra user effort there is a quick and much easier way to determine whether an interaction component may perform ineffectively. The next section shows how this indicator works and its limitation.

Indicator for ineffectiveness The calculation of the indicator, called the *Standardised Reception Coefficient* (SRC), basically is the same as the calculation of the extra user effort, although without the weight factors assigned to each message. The indicator is a ratio between two other ratios: the ratio between the real numbers of messages received and sent upwards, and the ratio between the number of messages received and sent upwards in the case of optimal performance (Equation 3.1). Interaction components with an SRC of 1 suggest optimal performance, with an SRC below 1 suggest inefficiency, and with an SRC above 1 suggest an ineffectiveness problem, which often is accompanied by a reduction in performance as well. Although the SRC can reflect inefficiency, its benefit lies in predicting whether messages are *unintentionally* sent upwards (ineffectiveness). The indicator is based on the assumption that messages sent upwards that were established with fewer messages than required when the task is optimally executed are probably the result of a usability problem related to the component itself and were probably not needed to complete the task. Therefore, SRC should not be regarded as a definition for ineffectiveness but as an indicator, because it tries to tell whether users send messages upwards unintentionally. Furthermore, the relationship between SRC and ineffectiveness is a logical implication, which means that users can have sent messages upwards unintentionally without the SRC having a value greater than 1. However, if the SRC has a value greater than 1, the prediction is that messages were sent upwards unintentionally.

$$\text{SRC} = \frac{\text{Sent upwards}_{\text{observed}} \times \text{Received below}_{\text{optimal}}}{\text{Received below}_{\text{observed}} \times \text{Sent upwards}_{\text{optimal}}} \quad (3.1)$$

Calculation example of SRC With the number of messages received and sent upwards—both optimal and observed—the SRC value is calculated for the circle example of the drawing application (Table 3.5). The SRC of the Selection interaction component is below 1, indicating reduced performance. The real user applied a less than efficient manner to select the objects. The SRC of the RMBM interaction component is 1, indicating optimal performance. The real user activated the property window in an optimal manner. The SRC of Properties interaction component is above 1, indicating ineffectiveness and probably reduced performance. The real user unintentionally changed the properties of a circle twice because of a problem with setting the right properties. The SRC of the VDO interaction component is 1, suggesting optimal performance. However, the real user selected the polygon unnecessarily, which makes the performance sub-optimal. This is not reflected in the SRC because of the additional property setting. Ineffectiveness of the lower-level interaction component Properties makes the SRC of the higher-level interaction component less interpretable.

Table 3.5: The number of messages received and sent upwards in both optimal and observed task executions; next, the corresponding SRC and their interpretation.

	Received below		Sent upwards		SRC	
	Optimal	Observed	Optimal	Observed		
Selection	1	4	1	2	0.5	inefficient
RMBM	2	4	1	2	1	optimal
Properties	6	10	1	2	1.2	ineffective
VDO	2	4	1	2	1	violation

Limitation SRC Calculating the SRC for all the messages of an interaction component demands that the optimal performance and the observed performance have similar ratios between the messages type that are sent upwards. If this is not the case, separate SRC should be calculated for each message type sent upwards by interaction component. A small example in which this requirement is not met may help to make this limitation more clear.

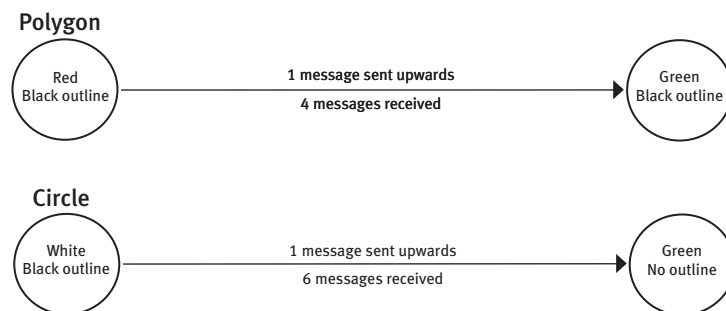


Figure 3.9: A state diagram of setting the properties of the polygon and the circle. The Properties interaction component will receive 5 messages on average per message sent upwards, when the task is optimally executed.

The objective is to calculate the SRC value for the Properties interaction component in the drawing application, but this time for another task. At the start of the task, the polygon is red and has a black outline. The task is to set the colour of the polygon and circle to green, and to remove the circle's black outline. If the task is optimally executed, the Properties interaction component receives ten messages: four for setting the polygon's properties and six for setting the circle's properties (Figure 3.9). Furthermore, the interaction component sends two messages upwards: one for the polygon and one for the circle.

Now, a 'real' user performs this task, who has a problem with making a distinction between the circle and the polygon; initially, the real user set the circle's property according to the

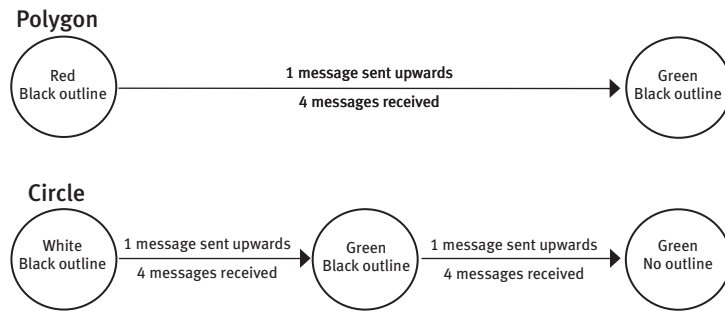


Figure 3.10: A state diagram of setting the properties of the polygon and the circle by a ‘real’ user, who initially takes the circle for a polygon. The Properties interaction component will receive 4 messages on average per message sent upwards.

desired polygon properties. As a result, the interaction component receives 12 messages: four for only setting the circle’s fill colour to green, four for removing the outline of the circle, and four for setting the polygon’s fill colour to green (Figure 3.10). The Properties interaction component also sent upwards one more message than in the optimal task performance. This means that in the optimal situation five messages are needed to send a message upwards on average, whereas in the real situation this is four. The SRC value Property interaction component is $(3 \times 10) (12 \times 2) = 1.25$, which should not be misinterpreted as an ineffectiveness problem since the interaction component operates optimally. Calculating an SRC value for two types of messages (setting one property or setting two properties at once) avoids this problem.

3.3.4 Subjective measures

Besides the performance measures, the perceived usability, scaled by subjects, can be used to evaluate the usability of the components. These component-specific questions are expected to be more sensitive than overall usability questions because they help the subjects to remember their control experience with a particular interaction component (Coleman, Williges, & Wixon, 1985). The difference between a component-specific and an overall questionnaire is that instead of the system, the name of the interaction component is used in each question. Besides the name of the interaction component, a description, a picture, or even a reference in the system of the interaction component can help to support the recollection.

Several questionnaires has been developed to determine the overall usability of a user interface (e. g. Chin, Diehl, & Norman, 1988; Davis, 1989; Gediga, Hamborg, & Düntsch, 1999; Hollemans, 1999; Kirakowski & Corbett, 1993; Lewis, 1995; Lin, Choong, & Salvendy, 1997). The five ease-of-use questions of the Perceived Usefulness and Ease-of-Use (PUEU) questionnaire (Davis, 1989) seems a suitable small set for a component-specific measure (Appendix B). They make no reference to the system’s appearance and are able to capture

well-formed beliefs developed by individuals about ease-of-use after only a brief initial exposure (Doll, Hendrickson, & Deng, 1998). The component-specific satisfaction questions are taken from the Post-Study System Usability Questionnaire (PSSUQ) (Lewis, 1995), one about how pleasant an interaction component was, and one about how much subjects liked an interaction component (Appendix B).

3.4 Comparison with other evaluation methods

3.4.1 Sequential data analysis

Often, in sequential data analysis, only lower-level events are recorded, which are first pre-processed before they are analysed. The three types of pre-processing are selection, abstraction, and re-coding (Hilbert & Redmiles, 2000). Selection involves separating low-level messages and sequences of interest from the ‘noise’. Abstraction involves relating lower-level messages to higher-level concepts. Re-coding involves generating new event streams based on the results of a selection and an abstraction process. The new event stream operates on a lower frequency band than the initial event stream. A frequency band is a range of frequencies in which messages are sent (Sanderson & Fisher, 1994). The concept of frequency bands in sequential data analysis can be seen as nothing more than a manifestation of the interaction layers in the user actions. Higher-level messages are sent over a longer time interval than lower-level messages because lower-messages are needed to create the higher-level messages.

In the proposed testing framework, the user’s control of an interaction component is analysed in the same frequency band that it operates. The analysis of higher-level interaction component comes down to more than only an analysis of pre-processed keystroke data. The higher-level messages are directly related to the control process of a higher-level interaction component. The compound messages, constructed from low-level messages afterwards, leave more room for discussion about the system interpretation of the lower-level messages and therefore lack a direct relation.

Extending the low-level messages log file with the system’s state makes it possible to construct the system interpretation of lower-level into higher-level messages. Still, it would require the analysis to envision the system response to a low-level message when the system is in a particular state. An example of such an approach can be found in the work of Lecerof and Paternò (1998). They keep track of the system state in what they call a Precondition Table, when they go through the user actions in a log file. This table is a set of temporal constraints that is automatically drawn from their task model; e. g. a user has to complete task A before beginning task B. When going through the actions in the log file, they keep track of the subtasks that are completed and preconditions that are satisfied. Simultaneously, they inspect whether preconditions are met for each action, and generate an error if not. The system responses are implicit envisioned in the formulation of the

task model because the task model requires an understanding of what system responses are acceptable to fulfil the user goal.

The analysis of temporal relations between the lower-level messages, which is often studied when dealing with sequential data (e.g. Markov analysis, lag sequential analysis, probabilistic finite state machine, Fisher's cycles, maximal repeating patterns, and regular expressions (Sanderson & Fisher, 1997)), becomes less important with the testing framework. The temporal relations are normally analysed in an attempt to construct a model with transition probabilities, or grammatical structure of messages. The grammatical structure facilitates the understanding of the higher-level interaction. However, with the recording of high-level messages this is no longer needed. On the other hand, analysing temporal relations within the messages received by a single interaction component and even combined with the states of the interaction component may give insight into why an interaction component operates as it does.

3.4.2 Not log-file-based usability evaluations

Other usability evaluation methods, such as thinking-aloud, cognitive walkthrough and heuristic evaluations, may in some cases be quicker in assessing the usability of an entire user interface. However, they suffer from a substantial evaluator effect in that multiple evaluators end up with different conclusions when evaluating the same user interface (Hertzum & Jacobsen, 2001). In the proposed testing framework, the systematic analysis that could even be automated leaves very little room for such an effect.

Furthermore, current usability evaluation methods have also received criticism for their ineffectiveness in finding real problems that lead to changes in a new version of the system (John & Marks, 1997). The introduction of component-specific usability measures is expected to increase the power of the evaluation method. These measures also lead designers unambiguously to the part that should be changed, because component-based evaluation directly links the usability assessment to a component.

This chapter describes a component-based usability testing framework. The framework gives rise to the idea of LPT as an approach to test components empirically. The benefits of the framework within the multiple versions testing paradigm lies in the statistical power of the measures, but also in the ability to study an individual component of the user interface directly and not just their impact on the overall usability. The latter is especially beneficial within the single version testing paradigm where the individual components can not be assessed on the base of overall usability measures. The following chapter describes an experimental evaluation of the testing framework. The claims made in this chapter will be put to the test there. The result of that experiment will help to position the framework with the set of other evaluation methods.

Chapter 4

An experimental evaluation of the component-based usability testing framework

4.1 Introduction

This chapter presents an empirical evaluation of the component-based usability testing framework described in the previous chapter. The testing framework gives shape to the claim that LPT can be employed to test the usability of interaction components. The theoretical discussion of the framework in the previous chapter boils down to a comparison of the power of component-specific and overall usability measures. The overall usability measures mentioned, such as keystrokes and questionnaire, are not new, in contrast with the proposed component-specific measures. The component-specific performance measures are based on the number of messages received by interaction components, whereas component-specific perceived ease-of-use and satisfaction measures are based on assisting users in the recall of their control experience of individual interaction components. Furthermore, the framework suggests a way to evaluate the usability of interaction component when only one user interface is considered. The testing framework proposes an impact assessment of the interaction components' usability on the usability of the entire user interface. Additionally, the framework offers an indicator to check whether, as a side effect, control loops disrupt performance measuring of higher-level interaction components. Table 4.1 presents a list of the claims, made within the testing framework. These five claims led to the design of an experiment and prototypes that were tested.

Table 4.1: Claims made in the component-based usability testing framework.

No	Claims
1	The objective component-specific performance measure is as powerful or more powerful than the objective overall performance measure based on the number of keystrokes.
2	Subjective component-specific usability measures are as powerful or more powerful than subjective overall usability measures.
3	The extra user effort to control an interaction component is positively correlated with the overall performance and negatively with the perceived ease-of-use and the satisfaction.
4	Ineffectiveness problems in an interaction component can influence the user's effort to control higher-level interaction components.
5	If an interaction component has an SRC value greater than 1, it operates ineffectively.

4.2 Method

The experiment was set up to compare different prototypes with variations in their usability. The use of predefined usability variations had to emphasise the validity of the usability measures. In this experiment, all usability variations addressed the complexity of dialogue structures that can be understood in terms of Cognitive Complexity Theory (CCT) (Kieras & Polson, 1985). This theory holds that the cognitive complexity increases when users have to learn more rules. Furthermore, the selected experimental design made it possible to study the effects that interaction components might have on each other. In other words, the robustness of the component-specific measures would be more clear, i. e. the impact of the other interaction components' usability. Furthermore, it should be possible to see how likely it is that interaction effects occur. An insight into the chance of an interaction effect makes it possible to speculate on the severity of ignoring an interaction effect in an experimental design. Beaudet and Williges (1988) showed that ignoring interaction effects can reduce the complexity of a study considerably and makes it possible to analyse large number of factors that may affect the usability of a system using a relatively small number of subjects.

The following section describes the prototypes that were used, the usability variations that were created, and the hypotheses that were drawn from the five claims. The design of the experiment and procedures are discussed in the subsequent sections.

Mail, Send Text Message, Read Text Messages, Read Address List, Edit Address List, Read Diary, Edit Diary, and Stand-by. The Function Selector processed subjects' input and constructed the request. Afterwards, the activation request was sent to the Telephone router, which passed the message on to the activated high-level interaction component. The high-level interaction component decided whether or not the request should be accepted. If so, this interaction component sent a message back to the Telephone router, asking it to send an activation message to the requested high-level interaction component and to direct future messages to this component. Another low-level interaction component, the Keypad, transformed key-presses on the keyboard of the mobile telephone into letters, numbers and cursor movements. Subsequently, the component sent the characters to the Telephone router that passed them on to the activated high-level interaction component. Finally, the three low-level interaction components labelled: Main Screen, Mode Screen and Menu Screen, were responsible for presenting the telephone feedback on the different parts of the telephone screen.

Function selection

To confirm the first claim, on the predictability of performance variation, two versions of the Function Selector were developed. This lower-level interaction component selected the functions of the mobile telephone, i. e. a menu. In one version of the Function Selector, the menu was relatively broad but shallow, i. e. all eight options available within one stratum (to prevent confusion the word stratum, instead of level, is used). In the other version, the menu was relatively narrow but deep, i. e. a binary tree of three strata. Users tend to be faster and make fewer errors in finding a target in broad menus than in deep menus (Snowberry, Parkinson, & Sisson, 1983). In terms of CCT, the deep menu structure requires subjects to learn more rules to make the correct choices when going through the deep menu structure.

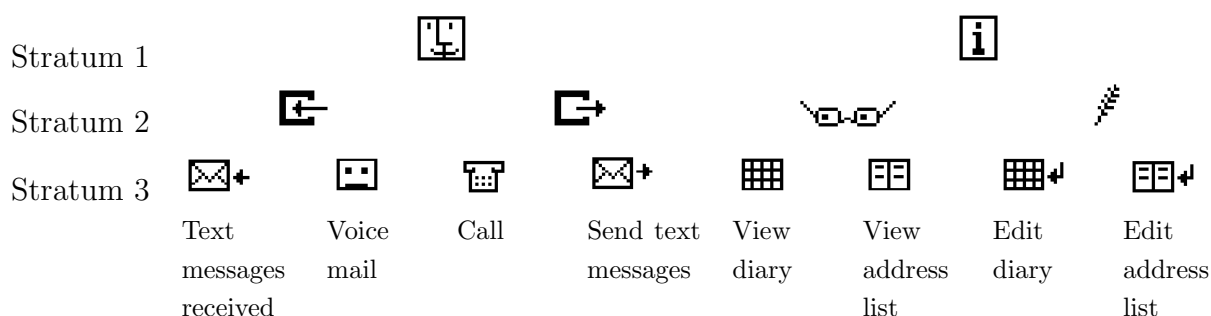


Figure 4.2: Menu icons of deep and narrow menu hierarchy.

Only the third stratum of Figure 4.2 was implemented in the broad/shallow menu structure. This menu was made up of eight buttons on the mobile telephone (Figure 4.3). Each button had an icon, which represented the function it activated. The narrow/deep menu



Figure 4.3: Implementation broad/shallow menu.

structure had a 2^3 structure, featuring all the strata of Figure 4.2. Figure 4.4 shows how the narrow/deep menu was implemented. This menu was activated after pressing the menu button. Subjects could browse through the menu with a Left, Right and Cancel button. The Left and Right buttons selected an option and took them one stratum deeper, whereas the Cancel button took subjects one stratum back, or deactivated the menu on the highest stratum. The lowest stratum presented the same icons as used on the buttons of the broad/shallow version (Figure 4.3). The choices on the higher strata were also presented with icons, however, they were designed to be vague (Figure 4.2, strata 1 and 2), having little association with the lowest-stratum options. Snowberry et al. (1983) suggested that this could increase the number of wrong choices on higher strata.

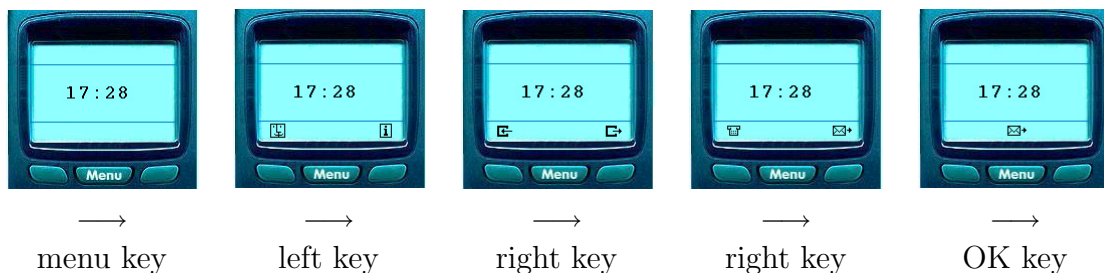


Figure 4.4: Menu options displayed on screen in the cases of the deep/narrow menu hierarchy. From left to right, sequence to activate the Send Text Message function.

In this experiment, images of the low-stratum icons or the (same) icons on the buttons were given in the subject's task description and on an instruction card. This eliminated a possible advantage of the broad menu structure. Otherwise, subjects could compare all options at once and pick the most suitable one in the broad menu structure, whereas in the deep and narrow menu structure, subjects would be faced with two choices at the

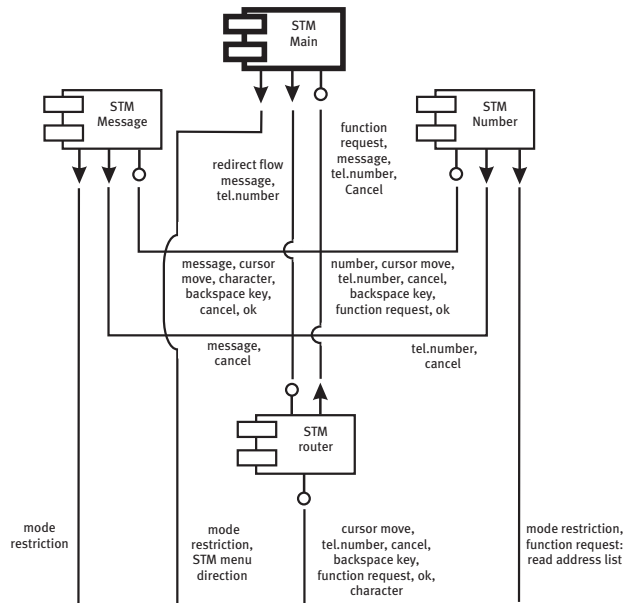


Figure 4.5: Internal structure of the Send Text Message interaction component.

lowest stratum. This would result in more unintended messages to be sent upwards in the deep/narrow version to higher-level interaction components, and therefore create an ineffectiveness problem. Another possible disadvantage of the deep menu structure was that subjects could mistake a low-stratum option that was not the target option for an option that led to another stratum in the menu and unintentionally send a message to a higher-level interaction component. To avoid this situation, subjects had to press an ‘OK’ button after selection of the low-stratum option to activate a higher-level function in the telephone. To conclude the section about the Function Selector, the hypothesis derived from the first claim, which focuses on the performance of a lower-level interaction component, is given below.

Hypothesis 1 *The number of messages received by the Function Selector is an equal or better predictor of the performance variation between narrow/deep and broad/shallow versions than the number of keystrokes.*

Send Text Message Function

Confirmation of the first claim, but this time applied to higher-level layers, was pursued by manipulating a high-level interaction component in the mobile telephone. Two versions of a sub-interaction component, called the Send Text Message (STM) Main, established a usability variation in a high-level interaction component. The sub-interaction component was imbedded in the STM component —a compound component responsible for sending

text messages. Figure 4.5 shows the organisation of the STM interaction component. Besides the STM Main interaction component, the following interaction components were also imbedded: STM Message, STM router, and STM Number. The STM Message interaction component was a so-called String component, responsible for the message. The STM Number component was responsible for establishing a telephone number. This could be done in two ways. First, subjects could directly enter a telephone number. Second, subjects could activate an address list and choose a person. The STM router took care of incoming messages and distributed them to the other sub-components. The STM Main interaction component was responsible for the sequence in which subjects could enter a message and a telephone number. It combined these pieces of information and established a higher-level message to send an entered text message to a particular telephone number.

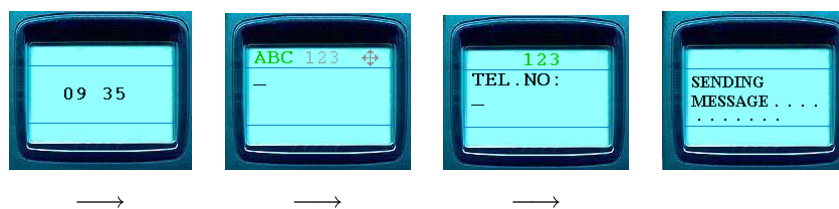


Figure 4.6: Sequence of telephone's screens when sending a text message with the simple, more usable, version.

In the more usable version, the so-called simple version of the STM Main, the system guided subjects through the required steps (Figure 4.6), whereas the less usable version, the so-called complex version, left the sequence of steps up to the subjects (Figure 4.7). This was meant to make subjects take more steps than minimally required as they searched for the right sequence. The simple version allowed subjects to directly enter a text message and the mobile telephone automatically requested the telephone number afterwards. In the complex version, subjects were first confronted with two options: to send the message or to edit it. The second option would lead subjects to two new options: edit the text message or the telephone number. For instance, when subjects chose to edit the text message, they could enter a new text or edit an unsent one. Afterwards, the subjects were brought back to the first two options, again requiring them to choose for the edit options and consequently in this case for the telephone number option. Only when both the text message and the telephone number were given, could subjects send the message with the send option. All these options were presented as icons that forced subjects to learn the icon-option mapping rules. Furthermore, they also had to learn in which order to choose the options.

The expectation was that subjects would not always write a text message or a telephone number without errors. This should create variance in the number of low-level messages. Consequently, the standard error might be greater than the difference in extra low-level messages that subjects sent to explore the options of the complex STM Main version. Analysis of the high-level messages should bypass the variance in the number of low-level



Figure 4.7: Sequence of telephone’s screens when sending a text message with the complex, less usable, version.

messages needed to enter a text message or a telephone number, because the whole text message and telephone number was regarded as one high-level message for the STM Main interaction component. Furthermore, each message to choose an option was regarded as an individual high-level message to the STM Main interaction component. This leads to the following hypothesis.

Hypothesis 2 *The number of messages received by the STM Main interaction component is an equal or better predictor of the performance variation between the simple and complex versions than the number of keystrokes.*

Keypad

The fourth claim is about ineffectiveness and its impact on a higher-level interaction component. An interaction component, called Letter, was manipulated in the experiment to study this claim. Figure 4.8 shows that this interaction component was embedded in the Keypad interaction component, together with the following other components: Mode, Keypad router, Cursor Movement, Number, and Keypad Off. The Letter interaction component transformed its input into letters. It received messages from the Keypad router

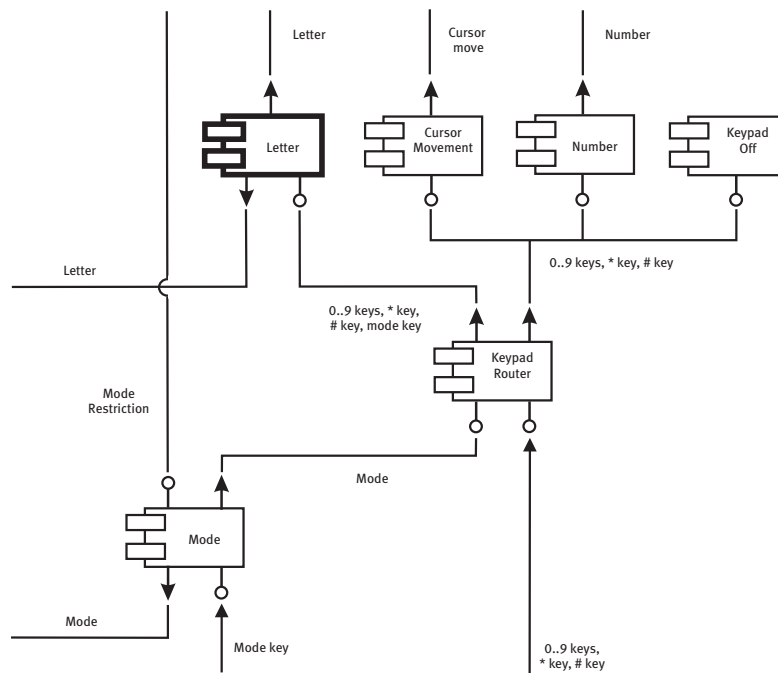


Figure 4.8: Internal structure of the Keypad interaction component.

and sent it to the Telephone router. The Keypad router determined which interaction component would receive the keyboard's input. Based on the input mode, the router decided whether the input would be interpreted as numbers, letters, cursor movements or to ignore the keyboard's input. The Mode interaction component handled the input mode. It received a 'mode change' request when subjects pressed the 'mode' button. High-level interaction components also sent it messages, instructing it which modes were allowed at that moment. For instance, when the telephone was expecting numbers for a telephone number, it was not allowed to switch to the letter mode.

The way subjects could enter letters combined with the likelihood of creating unwanted letters was used to create an effectiveness variation. The interaction component Letter had two versions. To enter letters, one version applied the Repeat-Key method, and the other version a Modified-Modal-Position method (Detweiler, Schumacher, & Gattuso, 1990). The Repeat-Key method involved having subjects press the key, containing the letter, the number of times corresponding to its ordinal position on the key (e.g., one time a '4' for 'G', Figure 4.9). The Modified-Modal-Position method involved having subjects first press either the '*' or '#' key, depending on whether the letter was in the left or right position on the button label and nothing when the letter was in the middle. This was followed by a press on the key containing the letter (e.g. "*" followed by '4' for 'G'). The Repeat-Key method is the easiest to instruct and to learn because users do not consider it intuitive to press a key associated with the position before they pressed the key with the letter (Detweiler et al., 1990). Furthermore, the Modified-Modal-Position method is more error



Figure 4.9: The mobile telephone’s keypad.

prone than the Repeat-Key method, i. e. more unwanted letters are created. Detweiler et al. argue that with the Repeat-Key method, there was only a simple rule to learn. This latter argument is in line with CCT.

The placement of the letters ‘Z’ and ‘Q’ on the keypad introduced an extra rule for the Modified-Modal-Position method. Originally, these letters were omitted in the North American Telephone keypad (Brodsky, 1991). If these letters are placed on the keypad, people prefer them on the 7 and 9 key (alphabetic order) instead of on the 1 key (Blanchard, Lewis, Ross, & Cataldo, 1993). In the experiment, they were positioned on the 1 key. This created ambiguity for the Modified-Modal-Position method, i. e. what key should precede a 1 key press? Subjects entering a ‘Q’ or ‘Z’ with the Modified-Modal-Position method, had to learn the additional rules that ‘Q’ should be considered as having the middle position and therefore did not require a press on a position key. The ‘Z’ required the subjects to press the ‘#’ because it was linked with the right position. Furthermore, the number of unintended letters with the Repeat-Key method was expected to reduce by giving subjects feedback on the letter currently selected as they pressed the key. Finally, a cursor automatically proceeded when the system received a letter. This was expected to help reduce some time-related ambiguity of the Repeat-Key method —whether the system relates a key-press to the succeeding key-presses or to a new letter.

The telephone’s String interaction components (e. g. STM Message interaction component) created a relation between the Letter component and the backspace key, causing possible ineffectiveness problems to spread. Messages from the Letter interaction component were received by these String interaction components. These interaction components were sub-interaction components of the following high-level interaction components: Send Text Message, Edit Address List, and Edit Diary. They were responsible for string manipulation, i. e. entering a new or editing an existing text message, name, or diary topic. Unintended letters, received by String interaction components, were expected to evoke new messages to counteract them. The backspace message was a typical message to undo previous created letters. When subjects pressed the backspace button a backspace message was directly sent to the Telephone router to be passed on to the currently active high-level interaction

component. In turn this high-level interaction component had to decide whether a String interaction component eventually handled the received backspace messages. Consequently, the number of backspace messages was related to the number of unwanted letters, which was related to the ineffectiveness problem of the Letter interaction component in this case. To study the effect of an ineffectiveness problem on a higher-level interaction component, a reference point is needed to see whether this effect is greater on the higher-level interaction component than on any other interaction component in the system. The Function Selector interaction component was taken as a reference point, since it was not a higher-level component of the Letter interaction component. Therefore, a third hypothesis was formulated:

Hypothesis 3 *The number of Backspace messages received by the String interaction components is an equal or better predictor of the effectiveness variations between the Repeat-Key and the Modified-Modal-Position versions of the Letter interaction component than the number of messages received by the Function Selector interaction component.*

Remaining claims

The eight mobile telephones also made it possible to formulate hypotheses for the other claims made in the testing framework.

Subjective component-specific measures The second claim was studied by comparing subjective overall measures with the subjective component-specific usability measure. Along with the questionnaire, a picture of the keyboard and the Function Selector implementation was given to help identify and remember the interaction components. In the questionnaire the following names were used to refer to the interaction components. The Function Selector interaction component was called the menu of the mobile phone. The Keypad interaction component was called the keyboard of the mobile phone. And the STM interaction component was called the text message function of the mobile phone. The hypothesis, to examine the second claim in the experiment, was formulated as follows.

Hypothesis 4 *The subjective component-specific usability measures applied to Function Selector, Keypad and STM Main interaction components are equal or better predictors of the usability variations between the versions of these interaction components than subjective overall usability measures applied to the mobile telephone.*

Objective component-specific performance measure in the single version testing paradigm The third claim about the ability to evaluate the usability of interaction component, when only one user interface is considered, can be examined with an analysis of the extra effort value of the Function Selector and the STM Main interaction components; both, as required, do not have ineffective lower-level interaction components. The following hypothesis could be formulated.

Hypothesis 5 *The extra effort value of the Function Selector and STM Main interaction components is positively correlated with the number of keystrokes and time; and negatively correlated with the perceived ease-of-use and the satisfaction.*

Ineffectiveness indicator The last claim about SRC holds a logical implication. It implies that if an interaction component has an SRC value greater than 1, some of its messages are *unintentionally* sent upwards by the user, or in other words, the interaction component operates ineffectively. The Modified-Model-Position version of the Letter interaction component was specifically designed with an ineffectiveness problem, the other version and other interaction components were not expected to be ineffective. Therefore, the following hypothesis was formulated to test whether SRC could detect this.

Hypothesis 6 *The Modified-Model-Position version of the Letter interaction component has an SRC value higher than 1; the Repeat-Key version of the Letter interaction component and the versions of the other interaction component have an SRC value not higher than 1.*

4.2.2 Design

All eighty participating subjects were students of Eindhoven University of Technology. An additional constraint was that the subjects did not use a mobile telephone on a daily or weekly basis, which was mentioned in the invitation, and confirmation was asked at the start of the experiment. Subjects were randomly assigned to eight groups that each operated on a different mobile telephone prototype. The subjects were told that only one prototype was tested, as would be the case in the single version testing paradigm. The kinds of tasks subjects had to perform with the mobile telephone were the making of a call to someone's voice-mail system; adding a person's name and number to the phone's address list; and sending a text message. The experiment consisted of three sessions, so that, learning effects could also be studied. In each session, subjects performed the three tasks. However, in each session they had to call someone else, add another name and number, and send a different text message; the computer randomly assigned the precise task instructions to the three sessions. All three text messages had the letter 'Z' and the number '1' in it. The three names, subjects had to add, held the letter 'Q'. No two consecutive letters in a text message or a name belong to the same key to avoid the timing-related ambiguity that might turn up with the Repeat-Key Method (Kramer, 1970). The call task was added to give subjects the impression that the whole mobile telephone was evaluated and not only some particular part of the mobile telephone. Task order effects were controlled by randomly assigning six group members to one of the six possible permutations of task types in a session. The remaining four members were again randomly assigned to one of the six permutations with the restriction that none of these subjects were assigned to the same permutation. The task order remained the same in the three sessions.

4.2.3 Procedure

At the beginning of the experiment, subjects were brought into a test room of a usability laboratory. They were told that the experiment was set up to evaluate the usability of a mobile telephone. To help them, they were given an instruction card that they could keep with them during the whole experiment. The instruction card explained the symbols on the telephone buttons and the icons of the menu's lowest stratum. Next, the subjects sat down behind the computer and the simulation application was started. The experimenter left the test room and went to the observation room. The application automatically assigned the subjects to a prototype and a task order. Furthermore, the application also assigned the task instructions to the sessions. In the next step of the experiment, subjects entered a training phase. They had to perform three training tasks with an alarm clock. In these tasks, subjects learned how to act in the experiment, i.e. to successfully fulfil the task as quickly as possible, which could require them to explore the user interface. They also practised that they had to click with the mouse on the buttons to interact with a prototype in this environment. Furthermore, the training showed the test procedure, such as reading an instruction on the screen, pressing a Start button to begin a task, and pressing a Ready button after completing the task. After the training phase with the alarm clock, the subjects performed the tasks with the mobile telephone in three sessions. Between the sessions, the subjects were asked to perform a filler task of solving simple equations for three minutes to study their ability to remember previous learning. At the end of the experiment, subjects were asked to evaluate the mobile telephone with the questionnaire on the computer. The computer gave the questions in a random order. After the experiment, the subjects received NLG 22.50 (€ 10.21) for their participation.

To prevent subjects from endlessly trying to solve a task, threshold times were set based on nine other subjects' performance in a pilot study. These times were calculated as the average (sub) task time plus three times the standard deviation.

4.3 Results

This section presents the results obtained from the experiment. The section is divided into the two testing paradigms. The multiple versions testing paradigm starts with the reliability of the questionnaire. This is followed by analyses performed on the performance measures of the three interaction components. Next, the perceived ease-of-use and satisfaction measures are analysed for an effect of the versions of the interaction components, followed by an overview of the robustness of the measures. Analyses for the learning effect conclude the analyses within this paradigm. In the single version testing paradigm, the SRC-values are first analysed and then the extra user effort values. But before results within these two paradigms are presented, a short explanation is given of what part of the data was analysed.

Table 4.2: Average subjects' characteristics and raw overall performance data.

Version interaction components	Age	Male	Female	Help ^a	Time ^b	Keys
Simple						
Repeat-Key						
Broad/shallow	20.9	9	1	0	13:23	408
Narrow/deep	22.1	7	3	0	20:43	609
Modified-Model-Position						
Broad/shallow	21.5	6	4	1	18:11	509
Narrow/deep	20.3	7	3	2	27:30	847
Complex						
Repeat-Key						
Broad/shallow	21.7	8	2	2	19:15	525
Narrow/deep	21.9	8	2	2	23:51	697
Modified-Model-Position						
Broad/shallow	22.3	7	3	0	18:15	506
Narrow/deep	20.7	6	4	0	26:43	771

^aThe number of subjects that received help. ^bMinutes : seconds.

4.3.1 Data preparation

The number of subjects that received help was relatively small and revealed no significant effect for the prototypes (Kruskal-Wallis test $\chi^2 = 8.50$, $df = 7$; $p. = 0.290$). However, the problems encountered, gave direction to how the data had to be analysed. Though the subjects were explicitly instructed to carry out the task on their own, six subjects asked for help because they got stuck, and one subject got help because the threshold time, after which intervention took place, was exceeded (Help column Table 4.2). One subject, out of the seven subjects that received help, was told to copy the name of the person precisely as stated in the instruction including one separating space. The subject put more spaces between the first and last name, to stop the name from being broken up over two lines. The subject explained that it would not look nice. This behaviour, of inserting additional spaces, was also observed among nine other subjects when they entered a text message. Furthermore, eight subjects inserted additional characters (e. g. a period) or words (e. g. an article or preposition). Therefore, the data analyses ignored possible extra spaces and characters, and only took the data until the subjects fulfilled the task for the first time; fulfilment meant: when the text messages was sent, when the call was made, or when the person was added to the address list. Fulfilling the task did not mean when the Ready button was pressed.

Table 4.3: Cronbach alpha derived from reliability analyses performed on the answers of the questionnaire.

Measure	Coefficient Alpha	
	Ease of use	Satisfaction
Overall mobile phone	0.85	0.90
Menu	0.87	0.75
Keyboard	0.85	0.86
Send text message function	0.89	0.81

4.3.2 Multiple versions testing paradigm

If a group of questions, such as the six ease-of-use questions or two satisfaction questions, measure the same underlying factor, it is easier to work with the average of the answers than with the individual answers. Therefore, before the data of the questionnaire were analysed for possible effects for the different versions of the interaction components, reliability analyses were performed first. The ease-of-use and the satisfaction questions had an acceptable reliability (Table 4.3) of more than the 0.7 – 0.8 minimal level often recommended (Landauer, 1997). The averages of the six ease-of-use questions and the two satisfaction questions were calculated for each interaction component and for the overall mobile telephone questions. From here on, these averages are taken as a perceived ease-of-use measure and as a satisfaction measure.

Function Selector

Before the power of the different measures can be compared, an effect for the version of the Function Selector interaction component on the measures has to be found. A MANOVA was performed on the seven dependent measures that are shown in Table 4.4. The analysis took the version of the Function Selector, the Letter and the STM Main interaction components as three between-subjects variables (two levels). In all measures, a significant effect was found for the version of the Function Selector interaction component. However, to perform all tasks optimally, 60 more keystrokes had to be made and 60 more messages had to be received with the narrow/deep version than with the broad/shallow version. To compensate for all a priori differences, another MANOVA was performed on two corrected measures —the number of keystrokes and messages received were compensated for the optimal performance difference in the various prototypes. Once more, the analysis found an effect for the version of the Function Selector interaction component in both measures (Table 4.5).

The first hypothesis is not about whether a difference between the two versions can be de-

Table 4.4: Results of multivariate and univariate analyses of variance with the version of the Function Selector as independent between-subjects variable.

Measure	Mean		Hyp.	Error	F	p	η^2
	Broad	Deep	df	df			
Joint measure	–	–	7	66	37.47	<0.001	0.80
Time in seconds	947	1349	1	72	29.56	<0.001	0.29
Number of keystrokes	461	686	1	72	37.72	<0.001	0.34
Number of messages received	67	265	1	72	155.34	<0.001	0.68
Ease of use mobile phone	5.5	4.8	1	72	11.86	0.001	0.14
Ease of use menu	5.6	4.5	1	72	22.33	<0.001	0.24
Satisfaction of mobile phone	4.4	3.8	1	72	4.25	0.043	0.06
Satisfaction of menu	4.6	3.5	1	72	15.96	<0.001	0.18

Table 4.5: Results of multivariate and univariate analyses of variance with the version of the Function Selector as independent between-subjects variable on measures corrected for the difference in optimal performance.

Measure	Mean		Hyp.	Error	F	p	η^2
	Broad	Deep	df	df			
Joint measure	–	–	2	71	60.96	<0.001	0.63
Number of keystrokes	437	602	1	72	20.27	<0.001	0.22
Number of messages received	52	190	1	72	75.36	<0.001	0.51

Note. In case of optional task execution, there existed a difference of 60 keystrokes and 60 messages received between the versions.

Table 4.6: Results of discriminant analyses with version of Function Selector interaction component as a dependent variable.

Independent variable	Correctly classified
Number of keystrokes	77.5 %
Number of messages received	96.3 %
Corrected number of keystrokes	68.8 %
Corrected number of messages received	88.8 %

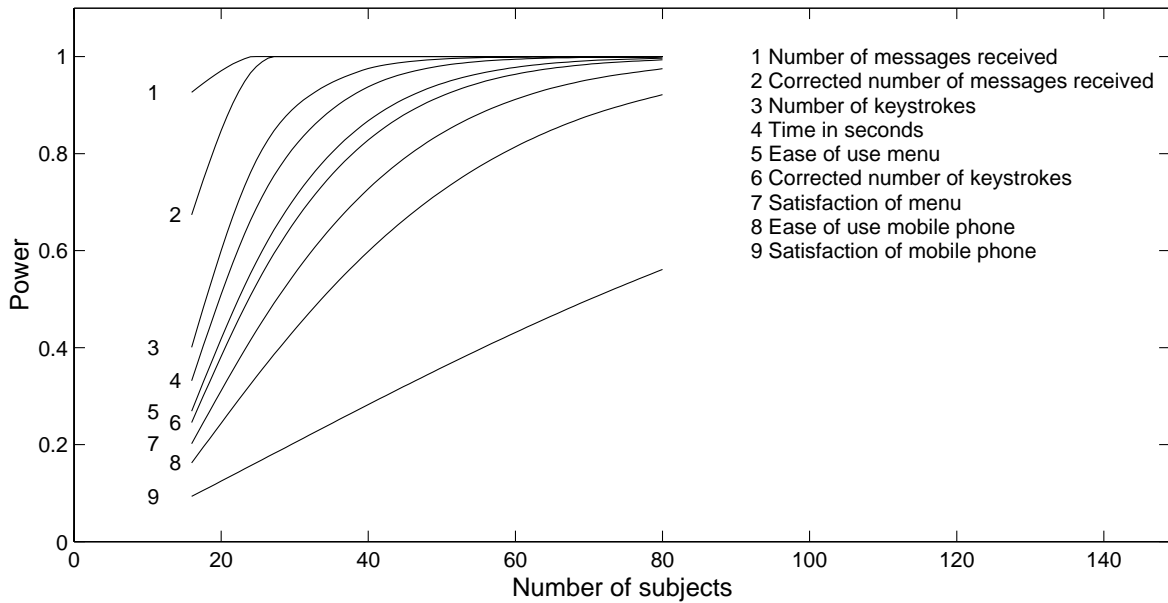


Figure 4.10: Probability that a measure finds a significant ($\alpha = 0.05$) effect for usability difference between the two version of the Function Selector interaction component.

tected, but how well this difference can be detected. Therefore, the power of the measures was analysed by calculating the probability of finding the difference as a function of the number of subjects. Figure 4.10 illustrates the chance of finding the difference between the two versions with a particular measure if this experiment had been carried out with fewer subjects. The figure shows the number of messages received as a more powerful measure than the number of keystrokes. Discriminant analyses were performed to study a related question —how well a measure could predict which version of the Function Selector interaction component a subject had interacted with. A discriminant analysis was conducted with the version of the Function Selector interaction component as dependent variable and the number of keystrokes, the version of the Letter interaction component and the version of the STM Main interaction component as independent variables. The latter two independent (control) variables were taken along to control for possible effects caused by variations between the different versions of these two interaction components. Table 4.6 shows the number of subjects that were correctly classified by the discriminant analyses. The discriminant analysis of keystrokes revealed that 77.5% of the subjects could be correctly classified. A significantly larger (Sign Test $n = 15$, $X = 0$, $p. < 0.001$) percentage of 96.3% was found by a discriminant analysis with the same set-up, except that the number of messages received replaced the number of keystrokes. To put the percentage into perspective, note that random allocation would classify 50% of the subjects correctly on average and a significant improvement in this experiment starts from 57.5% (Sign Test $n = 6$, $X = 0$, $p. = 0.031$). The discriminant analyses were repeated with the corrected measures. It showed 68.8% of the subjects could be correctly classified when based on the corrected number of

Table 4.7: Results of multivariate and univariate analyses of variance with the version of STM Main as independent between-subjects variable, and with the performance measures only related to the send text message tasks.

Measure	Mean		Hyp.	Error	F	p	η^2
	Simple	Complex	df	df			
Joint measure	–	–	7	66	18.16	<0.001	0.658
Time in seconds	523	672	1	72	8.15	0.006	0.102
Number of keystrokes	269	320	1	72	4.56	0.036	0.060
Number of messages received	12	49	1	72	74.18	<0.001	0.507
Ease of use mobile phone	5.0	5.3	1	72	1.15	0.288	0.016
Ease of use STM function	5.1	4.9	1	72	0.35	0.555	0.005
Satisfaction of mobile phone	3.9	4.2	1	72	0.93	0.339	0.013
Satisfaction of STM function	3.9	3.8	1	72	0.26	0.614	0.004

keystrokes. Whereas 88.8% correct classifications were made when based on the corrected number of messages received measure. A Sign Test found a significant difference ($n = 16$, $X = 0$, $p. < 0.001$) between these two percentages.

Send Text Message Function

In the previous section the claim was made that the complex version of the STM Main interaction component is harder to control than the simple version. Finding an effect for the version of the STM Main interaction component in the measures would validate this claim. A MANOVA was performed with the same three between-subject variables as before, but with the seven measures of Table 4.7 as dependent variables. Contrary to the previous analysis, the performance measures were taken only from the tasks in which subjects had to send a text message. In the other tasks, subjects did not have to control the STM Main interaction component, which therefore could increase the error term of especially the overall performance measures. Table 4.7 shows that the analysis found an effect for the version of STM Main only in the time, the number of keystrokes and the number of messages received. As before, a difference in the optimal task performance existed. At least 15 more keystrokes had to be made and at least 15 more messages had to be received in the case of the complex version (Figure 4.7). Table 4.8 shows the results of a MANOVA on the corrected performance measures. An effect for the versions was only found in the number of messages received.

The second hypothesis proposes a similar or lower power for the keystrokes than for the number of messages received in finding an effect for the version of the STM Main interaction

Table 4.8: Results of multivariate and univariate analyses of variance with the version of STM Main corrected for the difference in optimal performance as independent between-subjects variable. The measures were only related to the send text message tasks.

Measure	Mean		Hyp. df	Error df	F	p	η^2
	Simple	Complex					
Joint measure	–	–	2	71	20.85	<0.001	0.370
Number of keystrokes	249	289	1	72	2.30	0.134	0.031
Number of messages received	12	34	1	72	26.23	<0.001	0.267

Note. In the case of optional task execution, there was a difference of 15 keystrokes and 15 messages received between the versions.

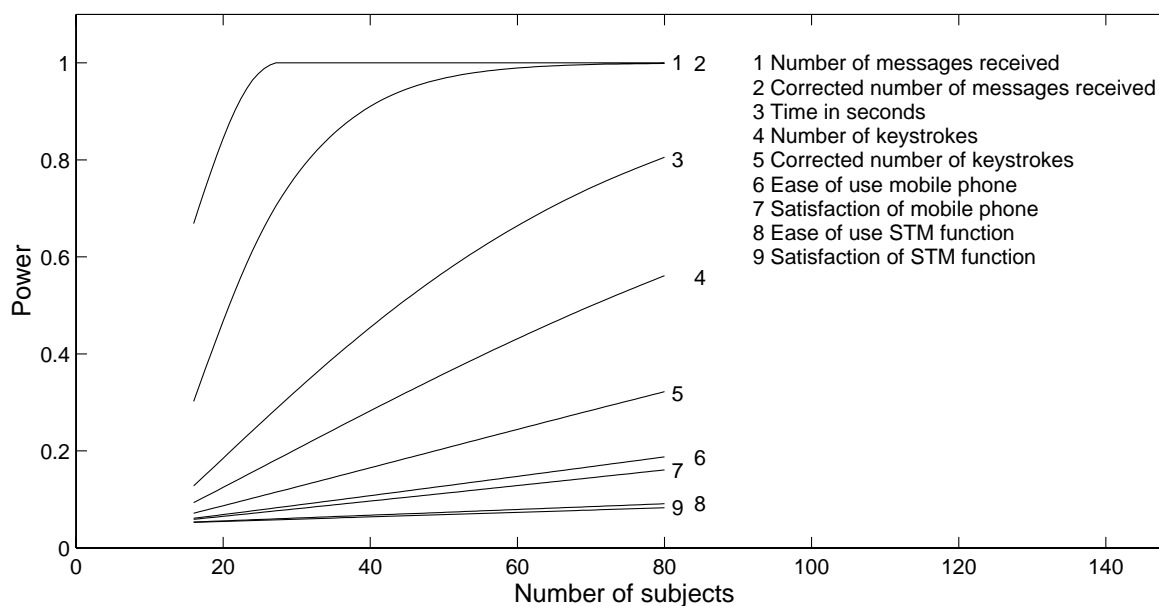


Figure 4.11: Probability that a measure finds a significant ($\alpha = 0.05$) effect for usability difference between the two version of the STM Main interaction component.

Table 4.9: Results of discriminant analyses with version of STM Main interaction component as a dependent variable.

Independent variable	Correctly classified
Number of keystrokes	56.3 %
Number of messages received	90.0 %
Corrected number of keystrokes	52.5 %
Corrected number of messages received	78.8 %

component. This was tested with power analyses that took the effect size of the measures out of tables 4.7 and 4.8. The results, illustrated in Figure 4.11, show that the measure number-of-messages-received was more powerful than the number of keystrokes, both as normal and as corrected. In addition, discriminant analyses were done with the STM Main interaction component as the dependent variable and with the versions of Function Selector and Letter interaction component as control variables. Table 4.9 shows the results of the discriminant analyses. One analysis showed a correct classification percentage of 90.0% when the number of messages received was used as independent variable. The other analysis took keystrokes as the independent variable. The results showed a significantly lower (Sign Test $n = 27$, $X = 0$, $p. < 0.001$) percentage of 56.3%. Once more, the two discriminant analyses were repeated with the corrected measures as independent variable. Correct classification based on the corrected number of received messages (78.8%) was significantly larger (Sign Test $n = 23$, $X = 1$, $p. < 0.001$) than the number of correct classification based on the number of keystrokes (52.5%).

Letter interaction component

The first analysis of the Letter interaction component looked for an effect of the versions in the different measures as was claimed in the previous section. A MANOVA was performed on the measures of Table 4.10 with the same three between-subject variables as before. The performance measures (i. e. time, number of keystrokes, and number of messages received) were only taken from the tasks in which subjects had to add a person to the address list or had to send a text message. A significant effect was found for the Letter interaction component versions in all measures, except in the ease-of-use and the satisfaction of the mobile telephone measures. Again, an a priori difference between the two versions existed when the tasks were performed optimally. To enter the text for a text message, the Repeat-Key required 37 keystrokes, whereas Modified-Modal-Position 38. However, the presentation of the analyses on corrected measures is neglected here because this difference is relatively small and the results are practically the same.

Could the effect for the version of the Letter interaction component be found in the users'

Table 4.10: Results of multivariate and univariate analyses of variance with the version of the Letter interaction component as independent between-subjects variable.

Measure	Mean		Hyp.	Error	F	p	η^2
	RK	MMP	df	df			
Joint measure	–	–	7	66	4.05	0.001	0.300
Time in seconds	872	1083	1	72	9.44	0.003	0.116
Number of keystrokes	438	537	1	72	10.34	0.002	0.126
Number of messages received	233	271	1	72	13.92	<0.001	0.162
Ease of use mobile phone	5.3	5.0	1	72	1.07	0.305	0.015
Ease of use keyboard	5.6	4.9	1	72	11.13	0.001	0.134
Satisfaction of mobile phone	4.3	3.9	1	72	1.76	0.188	0.024
Satisfaction of keyboard	4.6	3.8	1	72	8.97	0.004	0.111

Note. RK: Repeat-Key, MMP: Modified-Modal-Position.

Table 4.11: Results of multivariate and univariate analyses of variance with the version of the Letter interaction component as independent between-subjects variable

Measure	Mean		Hyp.	Error	F	p	η^2
	RK	MMP	df	df			
Number of Backspace messages	13	45	1	72	73.17	<0.001	0.504
Number messages received by Function Selector	102	112	1	72	0.65	0.424	0.009

Note. RK: Repeat-Key, MMP: Modified-Modal-Position.

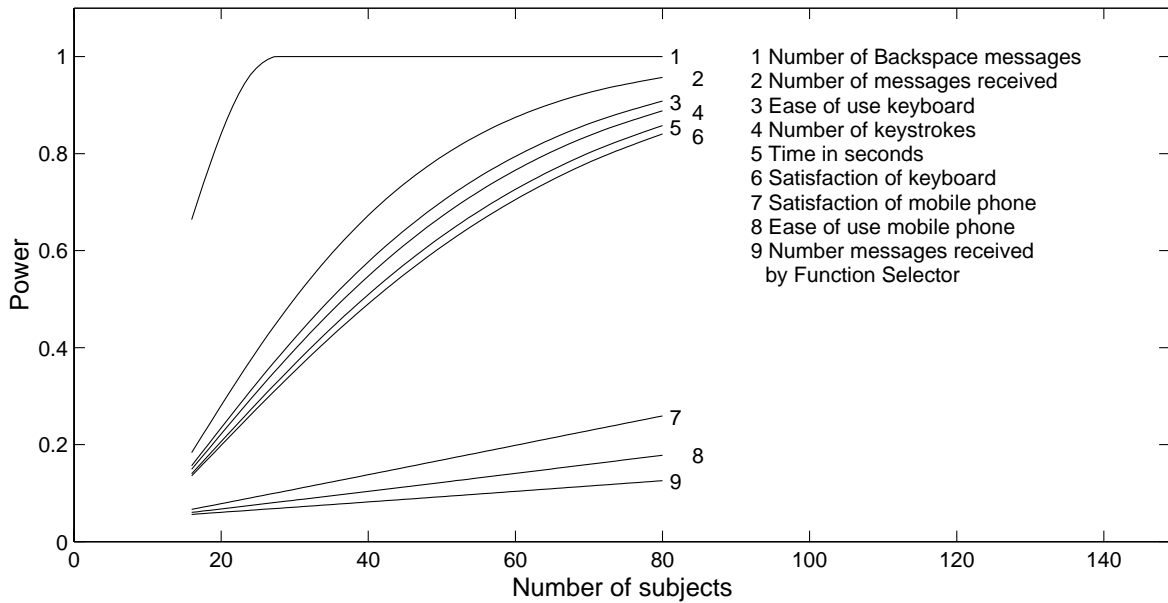


Figure 4.12: Probability that a measure finds a significant ($\alpha = 0.05$) effect for usability difference between the two version of the Letter interaction component.

control of other interaction components? This was the next question to be answered. An additional MANOVA was conducted with the dependent variables the number of backspace messages received by String interaction components (e. g. STM Message interaction component), and the number of messages received by the Function Selector interaction component as a reference point. Again, both measures were only taken from tasks in which subjects had to add a person to the phone’s address list or had to send a text message. An effect for the version of the Letter interaction component was only revealed in the number of backspaces (Table 4.11).

The following analyses tested the third hypothesis. Figure 4.12 shows the results of power analyses that used the effect sizes from of tables 4.10 and 4.11. The power of the number of backspace messages received by the String interaction components was higher than the number of messages received by the Function Selector interaction component. Two discriminant analyses were performed that had the version of the Letter interaction component as the dependent variable. Both took the version of the Function Selector and STM Main interaction components as control variables. One analysis showed that 81.3% of the subjects could be correctly classified when the number of backspaces was taken as the independent variable. The other analysis showed a significantly lower (Sign Test $n = 32$, $X = 6$, $p. = 0.011$) percentage of 56.3%, when the number of messages received by Function Selector interaction component was used instead of the backspaces.

Perceived ease-of-use and satisfaction

The above already showed that perceived ease-of-use and satisfaction measures could find an effect for the version of an interaction component in some cases. However, are subjective component-specific component measures equally or more powerful than subjective overall measures when it comes to finding these effects? The fourth hypothesis states this. Figures 4.10, 4.11 and 4.12 present the power of the subjective overall and the subjective component-specific questions, for both the ease-of-use and the satisfaction questions. It does give an ambiguous answer. Therefore, the hypothesis was put to the test. First, 12 (3 for the interaction components \times 2 for the measures' scope \times 2 for the usability dimensions) discriminate analyses were performed with the subjective overall or the subjective component-specific measures as an independent variable. All analyses took the versions of the remaining two interaction components as control variables. Second, the sum of correct predictions made by the subjective overall and by the subjective component-specific measures was calculated per subject resulting in a value between 0 and 3 (the number of times the subject was linked with a correct version). Third, the numbers of correct predictions made with the subjective overall and with the subjective component-specific measures were compared. A Wilcoxon Matched-Pairs Signed-Ranks Test revealed neither a significant difference in the case of ease-of-use questions ($n = 62$, $T = 30$, $p. = 0.907$), nor in the case of the satisfaction questions ($n = 61$, $T = 27$, $p. = 0.308$).

Robustness of the measures

More important in the single version testing paradigm than in the multiple versions testing paradigm is the robustness of a measure. The robustness of a measure is synonymous with its insensitivity to usability effects of other interaction components. The sensitivity can manifest itself in two ways. It can show up as an interaction effect or as a main effect for the version of another interaction component in the case of a component-specific measure. The MANOVA's mentioned earlier did not find any significant interaction effect in the performance measures. Nor did the analyses reveal a significant main effect for the other versions of interaction component in the component-specific performance measure. However, the MANOVA's did find other main effects and interaction effects in the ease-of-use and satisfaction questions. In the ease-of-use questions of the text message function, a MANOVA revealed a main effect ($F(1,72) = 4.14$; $p. = 0.046$) for the version of the Letter interaction component. In the ease-of-use questions of the keyboard, an analysis found two two-way interaction effects: one ($F(1,72) = 8.86$; $p. = 0.004$) between the version of the Function Selector and the STM Main interaction component, another one ($F(1,72) = 6.44$; $p. = 0.013$) between versions of the Function Selector and the Letter interaction components.

The MANOVA's on the satisfaction questions also found a two-way interaction effect and other main effects in the component-specific measures. As such, the analysis on the satisfaction questions about the keyboard revealed a main effect ($F(1,72) = 5.29$; $p. = 0.024$)

for the version of the STM Main interaction component, whereas the analysis on the satisfaction questions about the text message function revealed a main effect ($F(1,72) = 7.31$; $p. = 0.009$) for the version of the Letter interaction component. An analysis on the overall satisfaction of the mobile telephone found two interactions: one ($F(1,72) = 9.42$; $p. = 0.003$) between the version of Function Selector and Letter interaction components, and another one ($F(1,72) = 6.35$; $p. = 0.014$) between the version of the Function Selector and the STM Main interaction component. To conclude, an analysis on the satisfaction questions about the keyboard found a two-way interaction effect ($F(1,72) = 8.46$; $p. = 0.005$) between the version of the Function Selector and the STM Main interaction component.

Learning effects

Comparing the performance per sessions can show variations, which can be interpreted as learning or recall effects. Therefore, a doubly MANOVA with repeated measures was performed with the sessions as within-subject variable and the measures of Table 4.12 as multiple dependent variables. The measures included all types of tasks. The versions of the interaction components were taken as three between-subjects variables. Table 4.12 shows an effect for the sessions in all measures, both multivariate and univariate. Over the sessions all values decreased. The multivariate analysis also revealed two-way interaction effects between the sessions and the version of the Function Selector interaction components ($F(10,63) = 15.70$; $p. < 0.001$) and between the sessions and the version of Letter interaction component ($F(10,63) = 2.92$; $p. = 0.005$). These interaction effects were again found only in a univariate analyses of the overall measures and in the corresponding component-specific measures. However, a two-way interaction between the sessions and the version of the STM Main interaction effect ($F(1,72) = 13.81$; $p. < 0.001$) was only found in the univariate analysis of the component-specific measure of STM Main interaction component—the number of messages received by STM Main interaction component.

4.3.3 Single version testing paradigm

Ineffectiveness indicator

The SRC is suggested as an ineffectiveness indicator. A value above 1 should indicate an ineffectiveness problem, which was hypothesised to be the case only for the Modified-Model-Position version of the Letter interaction component. Table 4.13 shows the SRC values for the various versions of the interaction components. In all four prototypes, the SRC value of the Modified-Model-Position version was not below 1 and in three cases significantly greater than 1. None of the other interaction components, with the exception of the narrow/deep of the Function Selector, have a value above 1. The SRC of the Function Selector interaction component did not meet with the expectations, and therefore the SRC was analysed in more detail. Higher-level interaction components request disproportionately more *Ok* and *Cancel* messages than the *Function Request* messages that needed more

Table 4.12: Results of multivariate and univariate analyses of variance with sessions as an independent within-subjects variable.

Measure	Mean in session			Hyp.	Error	F	p	η^2
	1	2	3	df	df			
Joint measure	–	–	–	10	63	44.00	<0.001	0.875
Time in seconds ^a	729	229	190	1	72	255.05	<0.001	0.780
Number of keystrokes ^a	307	139	128	1	72	114.21	<0.001	0.613
Messages Function Selector ^a	102	35	30	1	72	97.55	<0.001	0.575
Messages Letter ^a	110	73	70	1	72	55.81	<0.001	0.437
Messages STM Main ^a	16	7	7	1	72	20.49	<0.001	0.222

^aThe lower-bound is taken in the univariate tests because none passed the sphericity assumption test.

than one lower-level message to be created. This means a violation of the assumption that the optimal performance and the observed performance have similar ratios between the messages type that are sent upwards. Therefore, the SRC value was also calculated for *Function Request* messages. Consequently, none of these SRC values were above one. In the case of narrow/deep version, they were all significantly below one.

The SRC for String interaction components was also calculated. The results show the impact of ineffectiveness on a higher-level interaction component, for the String SRC was significantly below one in the third prototype —implying reduced performance. However, the third prototype violated the assumption that no directly lower-level interaction component sends more messages up than required since the Letter SRC is significantly above one. An ANOVA with the String’s SRC as dependent variable and with the version of the Letter interaction component as independent variable did not reveal ($F(1,78) = 3.71$; $p. = 0.058$) an effect for the version, and thus none for the ineffectiveness problem.

Extra user effort

The extra user effort was calculated for the same four interaction components per prototype over all sessions and types of tasks (Table 4.14). The calculation of the extra user effort was neglected when the SRC or the SRC of a directly lower-level interaction component was above one. Consequently, the extra user effort of the STM Main interaction component was also calculated for all prototypes. Although Letter interaction component was a lower-level interaction component of STM Main, it was not a direct one. Furthermore, the String interaction components did not seem to transport the ineffectiveness problem of the Letter interaction component upwards, because none of the SRC of the String interaction components was significantly above one.

After calculation, the extra user effort measure was validated by correlating it with other

Table 4.13: The interaction component's SRC values for the different prototypes.

Version interaction components	Function Selector				
	All	Fun.req.	Letter	String	STM Main
Simple					
Repeat-Key					
Broad/shallow	1	1	0.97*	0.87**	0.96*
Narrow/deep	1.34**	0.58**	0.98	0.87	0.84**
Modified-Model-Position					
Broad/shallow	1	1	1.06**	0.74**	0.84*
Narrow/deep	1.20**	0.46**	1.03	0.80**	0.79**
Complex					
Repeat-Key					
Broad/shallow	1	1	0.97*	0.97	0.63**
Narrow/deep	1.37**	0.53**	0.97*	0.98	0.76*
Modified-Model-Position					
Broad/shallow	1	1	1.03**	0.90	0.66**
Narrow/deep	1.34**	0.49**	1.03*	0.94	0.54**

Note. H_0 : value = 1. Fun. req. : function request.

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

Table 4.14: The interaction component's extra user effort in keystrokes for the different prototypes.

Version interaction components	Function			STM Main	
	Selector	Letter	String	Normal	Conser. ^a
Simple					
Repeat-Key					
Broad/shallow	0	14.2**	20.3*	2.2	0.4*
Narrow/deep	47.7**	12.5	15.3	13.6*	2.5*
Modified-Model-Position					
Broad/shallow	0	–	–	22.8	0.7
Narrow/deep	76.3**	4.6	16.2	22.1*	5.8
Complex					
Repeat-Key					
Broad/shallow	0	15.3	0.2	33.6	18.0*
Narrow/deep	61.6**	14.9**	0.7	39.5	25.5
Modified-Model-Position					
Broad/shallow	0	–	–	34.1**	14.1**
Narrow/deep	69.4**	–	–	73.3**	44.1*

Note. H_0 : value = 0.

^aAccording to the more conservative approach.

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

measures. The fifth hypothesis states that the extra effort value of the Function Selector and STM Main interaction components would positively correlate with the overall performance and negatively with the perceived ease-of-use and the satisfaction. Table 4.15 shows the partial correlations between measures for the Function Selector and for the STM Main interaction components. All correlations were controlled for the versions of the other two interaction components. The correlations with the extra user effort of the Letter interaction component were refrained from because three prototypes had a Letter SRC above one. Only significant partial correlations, with the hypothesised direction, were found between the extra user effort of the Function Selector interaction component and the other measures. Furthermore, all measures had a significantly partial correlation with the extra user effort of the STM Main interaction and with hypothesised direction; except for the correlations with the mobile telephone satisfaction, they did not reach a significant level.

The final analyses focused on the validity of applying the extra user effort to order interaction components within a prototype for their improvement potential. The validity was checked by comparing the arrangement of the interaction components ordered by the extra user effort values with that of the arrangement order by other usability measures that had a common unit in which a measure was expressed. The comparison was done on the difference between the versions of the interaction component, because the overall measures did not give the improvement potential of a particular version of an interaction component. The arrangements that were compared were between the versions of the Function Selector and the STM Main interaction component. The difference of the version of the Function Selector was divided by the difference of the version of the STM Main interaction components (Table 4.16). A result between -1 and 1 would indicate that a higher improvement potential was gained between the version of STM Main interaction component. A result less than -1 or greater than 1 meant a higher improvement potential was gained between the version of the Function Selector interaction component. The latter situation was found in the extra user effort measure as well as in all the other measures.

4.4 Discussion

The result of the experiment shows no indication of pre-selection in the subject's ability to operate the mobile telephone. All subjects in the experiment were able to complete the tasks. The seven subjects that received help were not located in specific conditions. If this were the case the experiment would suffer from pre-selection —only the best subjects would be located or remain in one condition against all kinds of subjects in the other conditions. In such a situation, it would be better to take the same number of best-performing subjects in all other conditions to obtain a homogeneous sample. However, the drawback is that this sample would only represent the best-performing people and ignore the rest of the population.

Table 4.15: Partial correlation between extra user effort of particular interaction component and other usability measures.

Measure	Function	STM Main	
	Selector	Normal	Conser. ^a
Keystrokes	0.72**	0.45**	0.46**
Keystrokes corrected	0.64**	0.44**	0.44**
Time	0.63**	0.39**	0.45**
Overall ease-of-use	-0.43**	-0.26*	-0.28*
Component-specific ease-of-use	-0.55**	-0.34**	-0.36**
Overall satisfaction	-0.25*	-0.22	-0.10
Component-specific satisfaction	-0.41**	-0.37**	-0.30**

Note. All correlations were controlled for effects of the versions of the other interaction components.

^aAccording to the more conservative approach.

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

Table 4.16: The average difference between the versions of the interaction components and the impact ratio between the different interaction components.

Measure	Function Selector	STM Main	Priority ratio ^a
Extra effort conservative	64	23	2.8
Extra effort less conservative	64	30	2.1
Keystrokes	225	37	6.1
Keystrokes corrected	165	22	7.6
Time	402	135	3.0
Overall ease-of-use	0.8	-0.2	-3.2
Component-specific ease-of-use	1.1	0.2	7.2
Overall satisfaction	0.6	-0.3	-2.1
Component-specific satisfaction	1.1	0.2	7.3

^aPriority ratio = Function Selector / STM Main.

4.4.1 The power of the objective component-specific performance measure

The first claim about the power of the objective component-specific performance measure over the objective overall performance measure such as keystrokes was confirmed, both for lower and higher-level interaction components. The number of messages received in the case of Function Selector and the STM Main interaction component was a more powerful measure than the number of keystrokes. However, the broad/shallow version of Function Selector interaction component was not an interaction component according to the definition. It had no state that subjects could change. Subjects were in fact changing the state of the Telephone router —changing which telephone function would be activated and would receive the future higher-level messages. Then again, the subjects had the same objective when operating the narrow/deep version. Therefore, the analyses can also be seen as analyses on the messages received by the combination of Function Selector and Telephone router, which leads to the same conclusions. The results also show that the number of messages received is not always a more powerful measure. In the case of the Letter interaction component, its power was similar to the number of keystrokes. Therefore, evaluators should use both measures in a multivariate analysis in future usability tests.

4.4.2 The power of subjective component-specific usability measures

A component-specific questionnaire was not found to be more powerful than an overall questionnaire. The component-specific and the overall questionnaire showed similar results in finding a difference between the version of the interaction components. Therefore, in the multiple versions testing paradigm an overall questionnaire would be more favourable, because in that case fewer questions are needed in a usability test. Nevertheless, the question remains, whether an overall questionnaire works as well if subjects are not continuously reminded of the different interaction components of the user interface, as was the case in this experiment. The component-specific questionnaire might have performed better if the description of an interaction component is more precisely and more strongly linked with the recall of the control experience of the interaction component. In the single version testing paradigm, an overall questionnaire is not very helpful in evaluating an interaction component. In that case, a component-specific questionnaire is more suitable. The component-specific questionnaire in this experiment showed to be both reliable and valid but not robust in all cases. With the component-specific ease-of-use questionnaire, subjects gave an answer that was influenced by the usability of the other interaction components. It is debatable whether subjects did this because they thought that the usability of one interaction component was in their opinion affected by the other, or because the question also reminded the subject of the control experience of the other interaction component. Since subjects were not found to be influenced by the combination of two interaction components

when answering an overall ease-of-use questionnaire, but were found to be influenced in the case of the component-specific questionnaire, the latter explanation is more plausible for the ease-of-use questions. This can not be concluded for the satisfaction questions because the combined implementation of the versions of the Function Selector and STM Main interaction components was found to effect the satisfaction of both the keyboard and the mobile telephone. Besides that, the experiment was designed to manipulate the ease-of-use. No claims were made in advance about the satisfaction. Therefore, cautiousness should be practised when making conclusions based on the component-specific satisfaction and the overall satisfaction because the difference found may be an artefact instead of having something to do with the subjects' satisfaction.

4.4.3 Objective component-specific performance measure in the single version testing paradigm

The results show a positive correlation between the extra user effort and the overall performance, a negative correlation between the extra user effort and the perceived ease-of-use, and a negative correlation between the extra user effort and the satisfaction —third claim. This validates the extra user effort measure as a component-specific usability measure. The ability to say something about the accuracy of the extra user effort assessment is limited in this experiment. For the version difference of the STM Main interaction component, the conservative method assessed an extra user effort of 23 keystrokes, whereas the overall number of corrected keystrokes came up with 22. A large deviation was found for the version difference of the Function Selector interaction component, 64 versus 165. An explanation for the latter difference is the inefficiency of higher-level layers, which may cause many more keystrokes on this low-level layer. The interaction components were also compared for their improvement potential, which was possible since user's effort for each interaction component is expressed in a common unit. The order based on the extra user effort was similar to the order based on the overall performance, the overall ease-of-use, the overall satisfaction, the component-specific ease-of-use, or the component-specific satisfaction. The distance between the ordering was in most cases greater than the extra user effort assessed. Again, the inefficiency of higher-level layers may have caused this.

4.4.4 Ineffectiveness problems

The ineffectiveness of a lower-level interaction component can influence the user's effort to control higher-level layers —fourth claim. This is confirmed because the difference between the versions of the Letter interaction component was more noticeable in the number of backspaces than in the number of messages received by the Function Selector. This finding shows that receiving more messages on a higher-level layer may be caused by usability problems of lower-level interaction components. Still, it does not justify the conclusion

that lower-level interaction components can influence the higher-level interaction component usability. Ineffectiveness is particularly problematic when it comes to the analysis of the extra user effort in the single version testing paradigm. The analysis can handle inefficiency; ineffectiveness remains a problem because of the assumption that messages are sent up as a higher-level layer request. Solving ineffectiveness problems can effect the overall performance in two ways. First, besides being not effective, the interaction component can also be inefficient when it received more messages than required. Second, higher-level interaction components may receive additional messages to counteract the unintended messages. If the ineffectiveness can not be taken away, designers should spend more energy in the design of the interaction component that has to undo or correct the unintended effects. Take for example the design of a usable correction function if it is impossible to improve the speech recognition module of an application.

Ineffectiveness indicator

Support for the SRC as ineffectiveness indicator comes from the finding that only the version Letter interaction component with the ineffectiveness problem had an SRC value above one. The results also show the risk of interpreting the SRC value wrongly. The SRC of the Function Selector's total message flow was above one. This should not be interpreted as ineffectiveness because one of the SRC assumptions was not met. Therefore, future evaluators should convince themselves that these assumptions are not violated in a usability test. They should be alerted to the following two situations: first, interaction components that send up messages of multiple message types, and second, interaction components that have ineffective lower-level interaction components. In the first situation the chance of a violation can be reduced by limiting the SRC to messages sent upwards of only one or a small group of message types. This is only possible, if on the reception side, messages of one message type (or group of message types) are required to send up only these messages, as was the case with the *function request* message type. In the second situation, the violation can not be overcome, but only be determined by inspection of lower-level layers SRCs. That lower-level layers can disrupt the SRC of higher-level layers is not clearly shown by the results since no significant effect ($p. = 0.058$) was found for the version of the Letter interaction component in the SRC of String interaction components.

4.4.5 General remarks

How likely is an alternative explanation that the variation in the number of messages received is just an artefact caused by the difference in optimal task performance? As subjects have to perform more actions, more actions can go wrong. The less usable versions of both the Function Selector and the STM Main interaction components required fewer messages than their more usable version when the task was optimally performed. The measure may work for these usability problems, but can it also be applied when there is no difference in

optimal performance? The answer seems to be yes because the measure did find a difference, with the correct direction, between the versions of the Letter interaction component. The versions required nearly the same number of messages when optimally performed. Furthermore, the learning or recall effect found supports this answer. Assuming that over time the subject became better at performing the task, the number of messages received decreased accordingly. The finding that over time the performance of the less usable versions decreased more than the more usable versions, is again supportive of interpreting the measure as a performance measure.

Another observation is the absence of interaction effects between the versions of the components in the objective measures, for the overall and the component-specific measures. The results of the experiment described in chapter 2 found no interaction effects between the different components either. The replication of this finding, but now with a more realistic user interface, makes the claim that components do not affect each other's usability more likely to be true. Still, there may be factors that are not considered in these experiments that cause the opposite.

4.5 Conclusions

The claim that LPT can be utilised to test the usability of interaction components is justified. The study shows that an objective component-specific performance measure is as effective or more effective in determining the interaction components' performance than overall measures in cases where components operate independently. Subjective component-specific usability measures were, however, not more powerful than subjective overall measures. Their benefit, as well as the component-specific performance measure, is their applicability within a single version testing paradigm, where overall measures can not help to locate usability problems in the user interface architecture. What still needs to be found out, however, is how well these measures can be applied in a usability test of larger applications, such as a text editor with a large function arsenal? In the single version testing paradigm, the drawback of component-specific performance measures is the recording of the message exchange between all interaction components, starting from the lowest-level layer, where effort values are later assigned to these elementary messages. In a multiple versions testing paradigm, the additional effort to gain component-specific performance measures is lower because the recording is limited to only the messages received by the interaction components under investigation. Employing the component-specific performance measures in real usability tests will provide more insight into how intrusive they are because special coding for logging is needed in the application. Future usability tests will ultimately also show if evaluators accept component-specific measures as proposed here, or reject them as too laborious and impractical.

This study shows that the ineffectiveness of lower-level interaction components can influence the user's effort to control higher-level interaction components. However, the study did not show that components affect each other's usability. The following chapters look

for factors outside the user interface architecture. This will be done within the multiple versions testing paradigm, since it allows for the study of interaction effect. However, the findings are also applicable to the single version testing paradigm. The component-specific testing framework opens the door to study these factors. Furthermore, it gives evaluators the opportunity to pinpoint troubling interaction components in a user interface.

Chapter 5

Effects of consistency on the usability of user interface components

5.1 Introduction

A rationale often used to predict the usability of a new user interface is the usability of the individual components. Developers employ usable ready-made components in their aim to create a usable system. However, when components are placed together, inconsistencies between them may diminish the components' usability. Components are developed to operate independently of each other. However, factors such as inconsistency can cause the components to lose their independence as the feedback of one component can affect the users' interpretation of the feedback from others.

Consistency has no meaning on its own; it is inherently a relational concept (Kellogg, 1989). Although users may establish the relation between the components, designers set the stage for possible confusion since they design the feedback. If inconsistency diminishes the component's usability, designers should create components that are consistent with one another. When deploying the components in new systems they should be attentive to possible inconsistency problems. Furthermore, when interpreting component-specific usability measures, testers should be aware that the cause of a decrease might be partially attributed to other components. The Layered-Protocol Theory (LPT), which incorporates components in its description of the user-system interaction, should also be studied in how it explains consistency relations.

5.1.1 Consistency and LPT

Consistency can be described as doing similar things in similar ways with agreement between agents about which things are similar (Reisner, 1993). This means that a component is regarded as consistent when both designers and users partition the interaction with the

component in the same way into sets of similar interactions. Furthermore, designers and users have to apply the same criteria, or dimensions, to consider the interactions with components to be similar. Likewise, inconsistency involves disagreement between designers and users about which things are similar, since what designers may find consistent may not be consistent for users at all (Grudin, 1989).

In terms of LPT, consistency is related to I- and especially to the E-feedback because designers of a system guide users in their action selection with E-feedback. When the E-feedback fits into the users' mental model, users can derive the consequence of an action from this mental model (Figure 5.1, the component-specific mental model). The E-feedback is also responsible for the users' activation of a mental model. However, if something else besides the interaction component's E-feedback were to determine what mental model users apply, the usability of an interaction component would be partially outside the control of its designer. Furthermore, a mental model might be applied to control other interaction components than the one that sent the E-feedback. This would mean that users apply a mental model to control a series of interaction components in a device (Figure 5.1, the general mental model). This would favour the use of an integral metaphor instead of a combination of multiple separate metaphors (composite metaphor) when applied in a user interface (Smilowitz, 1995). The metaphor should also properly fit the application domain since the application domain may also be partly responsible for the mental model that users apply (Smilowitz, 1995).

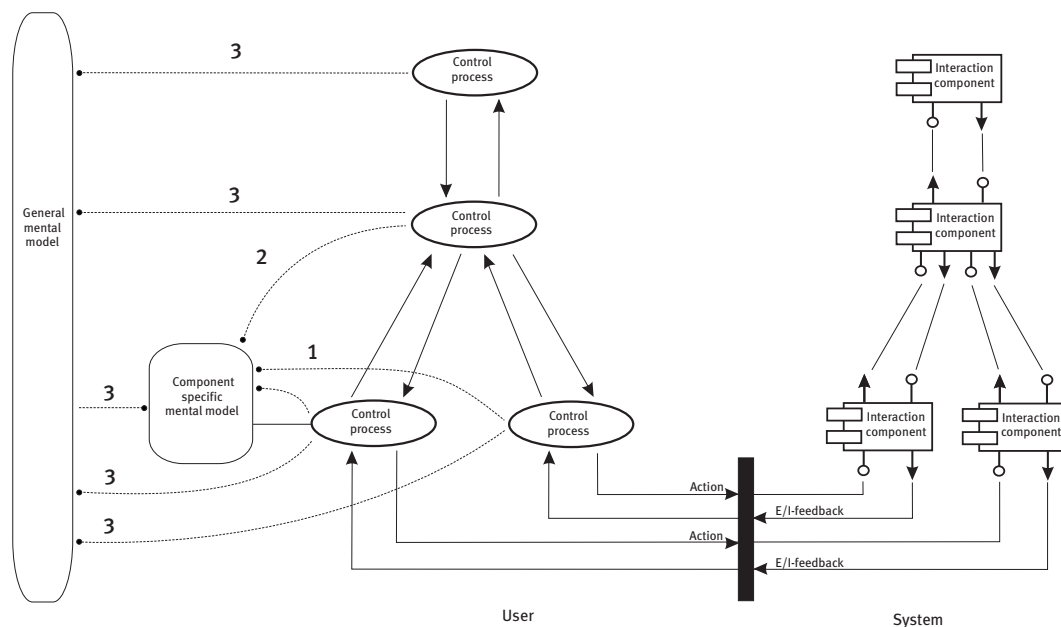


Figure 5.1: Layered interaction structure between user and system. On the user side, the relations between the control processes and the mental models are given. Numbers one to three present the three consistency relations that were studied.

Several studies have shown that consistency can affect the overall usability of a device (e.g. Payne & Green, 1989; Polson, 1988). However, little has been said about whether consistency can cause interaction components to affect each other's usability. This study looks at three situations (Figure 5.1) where this may occur: (Figure 5.1, relation 1) between interaction components in the same layer; (Figure 5.1, relation 2) between interaction components in different layers; and (Figure 5.1, relation 3) between an interaction component and an application domain. In all situations users misinterpret the E-feedback because of the mental model they apply. Why users apply a particular mental model may depend on factors outside the component, such as the feedback of other components or the application domain.

Before going into the three experiments that studied these three situations, the general experimental set-up of the experiments is presented. After the report of the experiments, the findings are discussed in general. The conclusions and further research directions are given in the final section.

5.2 General experimental set-up

All three experiments were conducted simultaneously under the control of one PC application written in Delphi 5. Each subject participated in all three experiments. The 48 subjects (32 male and 18 female), between 18 and 27 years old, were all students at the Eindhoven University of Technology and received NLG 15 (€ 6.81) for their participation. The subjects went through the following four phases: welcome phase, training phase, execution phase and the debriefing phase. In the welcome phase, subjects were brought into a test room of a usability laboratory. Here, they were told that the aim of the experiment was to test the usability of a number of consumer devices. Next, the subjects sat down in front of the PC and the application was started. The experimenter left the test room and went to the observation room. The application automatically assigned the subjects to three prototypes and a task order. In the training phase, subjects had to perform two tasks with a simulation of a clinical thermometer —switching it on, so it was ready to take someone's temperature, and displaying the last temperature measured. In these tasks, subjects familiarised themselves with the experimental procedure, i.e. to successfully fulfil the task as fast as possible, which still could require some exploration of the device. They also practised using the mouse to click on buttons in the picture (e.g. Figure 5.2) to interact with a prototype in this simulated environment. Furthermore, the training showed the test procedure, such as reading an instruction of the screen, pressing a Start button to begin the task, and pressing a Ready button after completing the task.

In the execution phase, the subjects performed tasks with three different devices in three sessions. In each session the subjects performed a task with each device. In between the sessions, the subjects performed a filler task. They heard a fairy tale and were asked to count the occurrences of two words in the tale. The order in which a subject had to operate a device was the same in each session. To control possible task/device order or

sequencing effects, the six permutations of the task/device order were equally distributed over the subjects. However, possible interaction effects between the task/device order and the experimental conditions were assumed to be zero. The design was also counter-balanced for possible two-way interaction effects between the conditions of the different experiments (e.g. the confrontation with one inconsistent user interface could make subjects more suspicious about inconsistency in another user interface). However, a possible three-way interaction between the conditions of the three experiments was again assumed to be zero.

Throughout the task performance, the message exchange between the interaction components of the devices was recorded. This made it possible to apply objective component-specific measures as described in chapter 3, i.e. the number of messages an interaction component received. In the debriefing phase, the subjects were asked to fill out a questionnaire (Appendix B) about the ease-of-use and satisfaction of the devices in general and some components in particular. As suggested in chapter 3, six ease-of-use questions were taken from the Perceived Usefulness and Ease-of-Use questionnaire (Davis, 1989) for each object and two satisfaction questions were taken from the Post-Study System Usability questionnaire (Lewis, 1995): one about how pleasant subjects considered something, and one about how much subjects liked something. This resulted in 64 questions that were presented to each subject in a random order. While answering questions, the subjects could see and operate the related prototype. Furthermore, a short description of the object subjects had to evaluate was presented in Dutch.

5.3 Experiment 1 —consistency within the same layer

The experiment to study the effect of consistency between interaction components within the same layer was conducted with four simulations of a room thermostat (Figure 5.2). The room thermostat had two very similar interaction components —daytime and nighttime temperature— which users presumably expected to be more or less similar things and therefore could be operated in a similar way.

5.3.1 Method

Figure 5.3 shows a part of the compositional structure of the room thermostat. To control one of the two temperature interaction components, subjects first pressed the Day or Night button (to the right of each display). This message was sent to the Router interaction component and resulted in the selection of one of the temperature interaction components, which was made visible by turning on a light. After this, the subjects could press the Left or Right button, which again sent a message to the Router interaction component. The Router interaction component passed this message on to the selected temperature interaction component, which adjusted its state accordingly.



Figure 5.2: An inconsistent room thermostat. The daytime temperature control is implemented with a moving pointer and fixed scale, and the nighttime temperature control with a fixed marker and a moving scale. The daytime temperature is selected, which is indicated by the illumination.

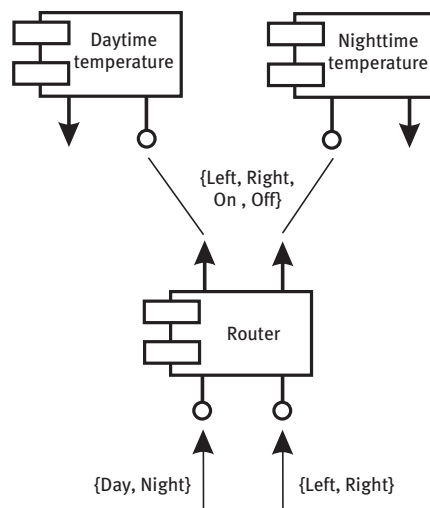


Figure 5.3: Part of the compositional structure of the room thermostat.

Two versions of both the Daytime and Nighttime Temperature interaction components were designed, which resulted in four prototypes. In one version of temperature interaction components, the temperature had a display with a moving pointer and a fixed scale (the upper temperature control in Figure 5.2), in the other version the display had a fixed marker and a moving scale (the lower temperature control in Figure 5.2). The Left and Right buttons had an opposite effect in the two versions. Where a click on the Right button in the version with a moving pointer would increase the temperature, the same actions in the version with a fixed indicator and moving scale would decrease the temperature.

The hypothesis was that the performance, the perceived ease-of-use and the satisfaction is higher when the subjects set the daytime and nighttime temperature with a prototype where both temperature interaction component are equipped with the same version, compared to different versions. The reason given for this effect is that when setting the temperatures, the subjects would apply the mental model constructed for the first interaction component to the other. When dealing with two different versions this approach would lead to mode errors (Norman, 1981), which the subjects subsequently had to recover from.

5.3.2 Results

Performance

A MANOVA was conducted on the performance measures: the keystrokes, the task time, the number of messages received by the Daytime Temperature interaction component, and the number of messages received by the Nighttime Temperature interaction component. The analysis took the version of the Daytime Temperature interaction component (2) and the version of the Nighttime Temperature interaction component (2) as between-subjects variables. The result of the analysis is presented in Table 5.1.

The multivariate analysis, presented here as the *joint measure*, found significant main effects for both the Daytime and the Nighttime versions. No significant two-way effect was found, although the p . value of 0.052 came close to the 0.05 α -level. To understand the direction of the effects and the effects on the particular parts of the user interface, the individual measures were also subjected to univariate analyses with the same between-subjects variables. The results of these analyses are also presented in Table 5.1.

The analysis of the overall measure task time only revealed a significant main effect for the Daytime version. Inspection of the mean showed that on average, subjects spent 76 seconds completing the three tasks when the Daytime Temperature interaction component was implemented with the moving pointer version, and 83 seconds with the moving scale version. The analysis of the other overall performance measure, the number of keystrokes, revealed no significant effects.

The analysis of the number of messages received by the Daytime interaction component revealed a significant main effect for the Daytime version. More messages were received

Table 5.1: Results of multivariate and univariate analyses of variance on the performance measures of the room thermostat for the independent variables Daytime version and Nighttime version.

Measure	Daytime version			Nighttime version			Daytime version \times Nighttime version		
	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>
Joint measure	4, 41	4.69	0.003	4, 41	3.97	0.008	4, 41	2.57	0.052
Task time	1, 44	4.10	0.049	1, 44	0.53	0.472	1, 44	0.61	0.438
Keystrokes	1, 44	1.43	0.238	1, 44	2.41	0.128	1, 44	1.09	0.302
Messages Daytime	1, 44	9.58	0.003	1, 44	0.05	0.833	1, 44	0.65	0.425
Messages Nighttime	1, 44	2.60	0.114	1, 44	9.22	0.004	1, 44	7.06	0.011

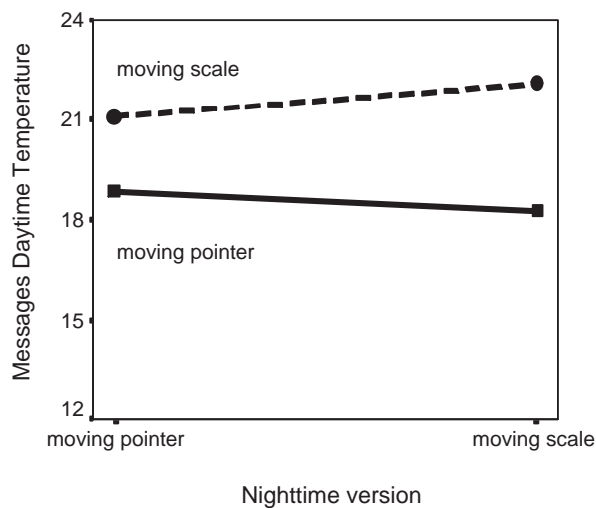


Figure 5.4: Number of messages received by the two versions of the Daytime Temperature interaction component. At least $3 \times (3 \text{ Left/Right messages} + 1 \text{ On message} + \text{possible } 1 \text{ Off message})$ were required to perform all tasks.

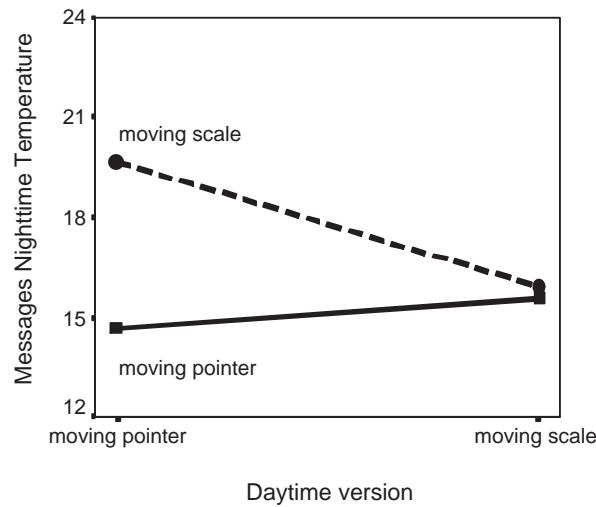


Figure 5.5: Number of messages received by the two versions of the Nighttime Temperature interaction component. At least $3 \times (3 \text{ Left/Right messages} + 1 \text{ On message} + \text{possible } 1 \text{ Off message})$ were required to perform all tasks.

when the Daytime Temperature interaction component was implemented with a moving scale than with a moving pointer control (Figure 5.4).

The analysis of the number of messages received by the Nighttime interaction component revealed a significant main effect for the Nighttime version. More messages were also received when the Nighttime Temperature interaction component was implemented with the moving scale than with the moving pointer control (Figure 5.5). In addition, the analysis found a significant two-way interaction effect between the Daytime and Nighttime versions. The explanation for this effect is that more messages were received in the prototype which had the moving pointer version for the Daytime Temperature and the moving scale for the Nighttime Temperature than in the other three prototypes (Figure 5.5).

Event pairs were studied to see whether the subjects set the nighttime temperature after they set the daytime temperature, or the other way around. Subjects had to switch from interacting with one interaction component to the other at least three times. On average, the subjects switched 3.0 times from sending messages to the Daytime to sending messages to the Nighttime interaction component. This was significantly ($t(47) = 55.44; p. < 0.001$) more than the other way around because on average, the subjects switched 0.2 times from sending messages to the Nighttime to sending messages to the Daytime interaction component.

Table 5.2: Cronbach alpha of the questionnaire items related to the room thermostat.

Measure	Description in questions	Coefficient Alpha	
		Ease of use	Satisfaction
Overall	Room thermostat	0.82	0.91
Daytime	Control of the day temperature	0.92	0.91
Nighttime	Control of the night temperature	0.92	0.94

Questionnaire

Questionnaire answers were analysed to see whether the versions of the temperature interaction components affected the subjects' perceived ease-of-use and their satisfaction. Table 5.2 shows the results of reliability tests performed on the answers of the ease-of-use and the satisfaction questions related to the room thermostat. All measures had an acceptable reliability above the 0.7 – 0.8 minimal level often recommended (Landauer, 1997). For ease of analysis, the averages of the six ease-of-use and the two satisfaction questions for the three objects (room thermostat and the Daytime and Nighttime Temperature interaction components) were taken as perceived ease-of-use and satisfaction measures. This reduced the 24 questionnaire answers for this experiment into 6 subjective measures.

Ease of use A MANOVA was conducted on the ease-of-use measure of the room thermostat, the Daytime Temperature and Nighttime Temperature interaction components. The analysis again took the Daytime (2) and Nighttime (2) versions as the between-subjects variables. The results are presented in Table 5.3.

The multivariate analysis (joint measure) found significant main effects for the Daytime version and the Nighttime version. The analysis also found a significant two-way interaction effect between the Daytime and Nighttime versions. Again, the individual measures were separately analysed (Table 5.3). The analysis did not find significant effects in the ease-of-use questions about the *room thermostat*.

Analysis of the ease-of-use answers about the *control of the day temperature* revealed a significant main effect for Daytime version. On a scale from 1 (low) to 7 (high), the subjects rated the ease-of-use of the daytime control higher if the component was implemented with the moving pointer version, instead of the moving scale version (Figure 5.6). The analysis also revealed a two-way interaction between the Daytime and Nighttime versions. Figure 5.6 shows that the combination of the Daytime Temperature implemented with the moving scale version and the Nighttime Temperature implemented with the moving pointer version led to a lower ease-of-use rating of the Daytime Temperature interaction component than for the other combinations. This explains the significant two-way interaction effect found.

Table 5.3: Results of multivariate and univariate analyses of variance on the ease-of-use measures of the room thermostat for the independent variables Daytime version and Nighttime version.

Measure	Daytime version			Nighttime version			Daytime version \times Nighttime version		
	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>
Joint measure	3, 42	7.43	<0.001	3, 42	9.81	<0.001	3, 42	3.49	0.024
Room thermostat	1, 44	2.30	0.136	1, 44	1.77	0.190	1, 44	2.90	0.096
Daytime	1, 44	11.52	0.001	1, 44	1.02	0.317	1, 44	4.18	0.047
Nighttime	1, 44	2.78	0.102	1, 44	20.24	<0.001	1, 44	7.73	0.008

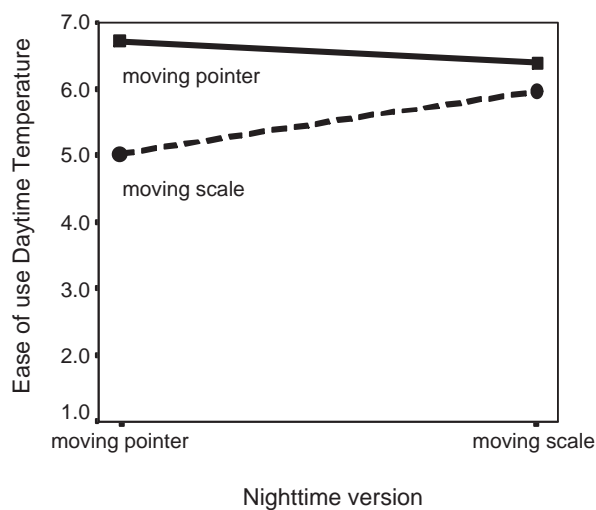


Figure 5.6: Ease-of-use rating of the two versions of the *Control of the day temperature*.

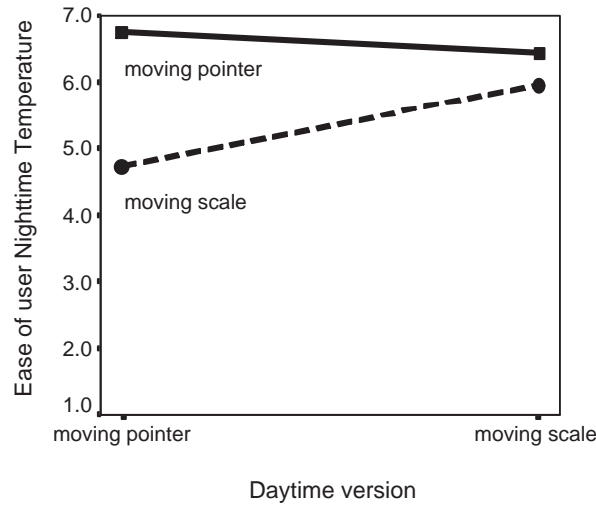


Figure 5.7: Ease-of-use rating of the two version of the *Control of the night temperature*.

The univariate analysis of the ease-of-use rating of the control of the night temperature revealed a significant main effect for the Nighttime version. For the case of the Nighttime Temperature, the subjects again rated the moving pointer version higher than the moving scale (Figure 5.7). The analysis found a two-way interaction between the two versions. The reason for the effect was that the subjects rated the control of the night temperature the lowest when the Nighttime Temperature was implemented with the moving scale and the Daytime Temperature was implemented with the moving pointer version.

Satisfaction A MANOVA was conducted on the satisfaction with the room thermostat, the Daytime Temperature and Nighttime Temperature interaction components. The analysis took the same between-subjects variables as in the previous analyses. The results are presented in Table 5.4.

The multivariate analysis (joint measure) revealed significant main effects for both the Daytime version and the Nighttime version. The analysis did not find a significant two-way interaction effect for the versions. Univariate analyses on the individual measures were also conducted (Table 5.4). The analysis of the satisfaction with the room thermostat revealed no significant effects.

The analysis on the rating of the satisfaction with the *control of the day temperature* found a significant main effect for the Daytime version. On a scale from 1 (low) to 7 (high), the subjects rated the satisfaction of the Daytime control higher if this component was implemented with the moving pointer, instead of the moving scale version (Figure 5.8).

The analysis on the rating of the satisfaction with the *control of the night temperature* found a main effect for the Nighttime version. The satisfaction with the night temperature was

Table 5.4: Results of multivariate and univariate analyses of variance on the satisfaction measures of the room thermostat for the independent variables Daytime version and Nighttime version.

Measure	Daytime version			Nighttime version			Daytime version \times Nighttime version		
	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>
Joint measure	3, 42	12.31	<0.001	3, 42	11.52	<0.001	3, 42	1.70	0.181
Room thermostat	1, 44	1.53	0.223	1, 44	1.30	0.260	1, 44	2.94	0.094
Daytime	1, 44	15.42	<0.001	1, 44	1.49	0.229	1, 44	3.43	0.071
Nighttime	1, 44	2.63	0.112	1, 44	18.12	<0.001	1, 44	4.12	0.049

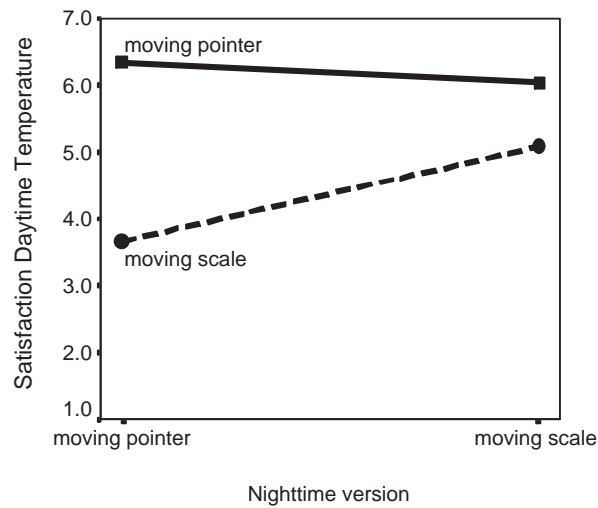


Figure 5.8: Satisfaction rating of the two versions of the *Control of the day temperature*.

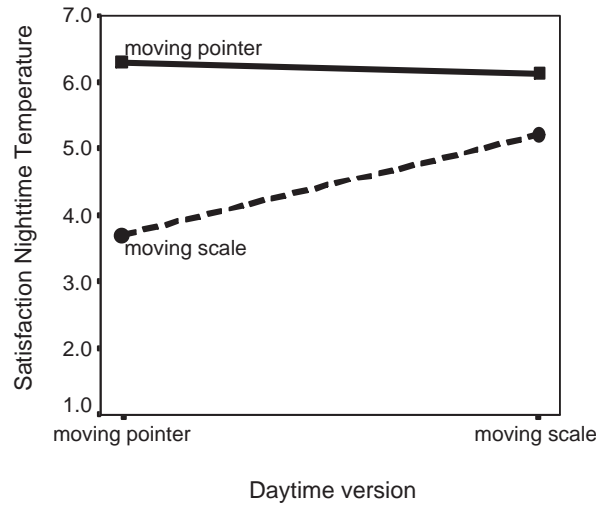


Figure 5.9: Satisfaction rating of the two versions of the *Control of the night temperature*.

rated higher when it was implemented as a moving pointer instead of a moving scale (Figure 5.9). Furthermore, the analysis found a significant two-way interaction effect between the Daytime and Nighttime versions. The low rating of the moving scale implementation of the night temperature, when the Day Temperature was implemented with the moving pointer version caused this interaction effect (Figure 5.9).

5.3.3 Discussion

The interaction effects found in the number of messages received by the Nighttime Temperature interaction component, in the ease-of-use and in the satisfaction rating of the night temperature control support the initially stated hypothesis. As hypothesised, the performance, the perceived ease-of-use and the satisfaction were higher when the subjects set the daytime and nighttime temperature with a prototype where both temperature interaction components were equipped with the same version than with different versions. This is again in line with the more general idea that inconsistency within one layer can reduce the usability of the interaction components.

The explanation for the significant interaction effects found may be that the mental model the subjects create when setting the daytime temperature was applied to understand the E-feedback of the Nighttime Temperature interaction component. This led to the performance of the wrong actions for the control of the Nighttime Temperature interaction component.

The significant interaction effects were always found in the component-specific measures of the Nighttime Temperature interaction component and only once in a component-specific measure (ease-of-use) of the Daytime Temperature. A preference in the task execution

seems to be the reason why interaction effects were primarily found in the component-specific measure of the Nighttime Temperature interaction component. The analysis of the event pairs showed that the daytime temperature was generally set before the nighttime temperature instead of afterwards. This preference in sequence stopped the mental model of the nighttime temperature from being activated when the subjects operated the daytime temperature. The reason why the subjects started with the daytime temperature might be the upper location of the daytime temperature display or because the instruction text first mentioned the daytime temperature, followed by the nighttime temperature.

Another observation was that the decrease in usability only occurred when the Daytime Temperature interaction component was implemented with the moving pointer version and the Nighttime Temperature interaction component with the moving scale version and not the other way around. One explanation for this is that the moving pointer version activated a more powerful (often or recently applied) mental model than that of the moving scale version, which consequently increased the likelihood that the E-feedback of the Nighttime Temperature interaction component was interpreted in the light of this mental model. The findings that the moving pointer version was more usable support the idea of a more powerful mental model.

An observation, not directly related to this experiment, but more to chapter 3, is the fact that the analyses of the overall measures only revealed a significant effect once, a main effect for Daytime version in the task time. This confirms the main hypothesis of chapter 3 that component-specific measures can be more powerful than overall measures.

5.4 Experiment 2 —consistency between lower-level and higher-level layers

The experiment to study the effect of consistency between interaction components in different layers was conducted with four simulations of a web-enabled TV set. A mistake that novice Lynx users probably easily make, served as a model for a possible inconsistency problem between two layers. Lynx is a text-based web browser that allows users to access the web in non-graphical environments without the use of a mouse. Users can select the links with the Up and Down arrow buttons on the keyboard. To activate the selected link, users have to press the Right arrow. With the Left arrow, users can return to the previous page. The possibility of an error may increase when links in the web page are placed on the same line. The supposed error occurs because of the activation of an inappropriate mental model —horizontal positioning with the Left and Right arrows. This example suggests that even although the Internet architecture is developed to make web pages independent from the browsers, users might run into trouble when on a higher-level layer the web-page server activate an inappropriate mental model for the interpretation of lower-level browser E-feedback.

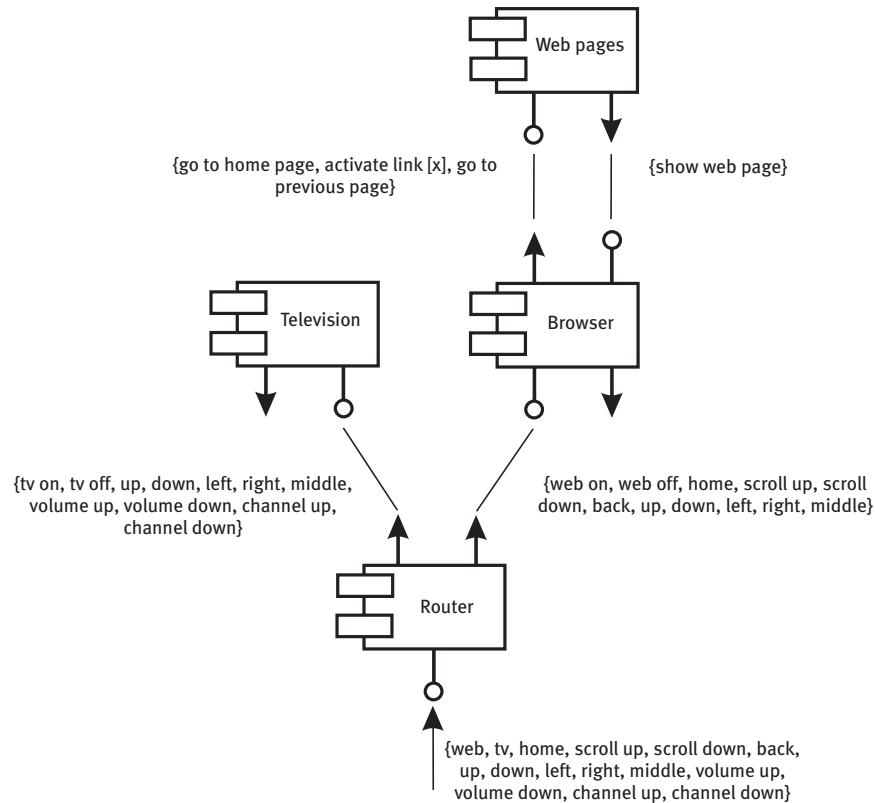


Figure 5.10: Part of the compositional structure of the web-enabled TV set.

5.4.1 Method

The tasks the subjects had to perform, using a web-enabled TV set, was to find the web page that gave the departure times of a bus based on the bus stop, the bus number, the city and the province, which were all given in the instruction. The home page of the web-enabled TV set simulation was a fictitious Dutch bus travel information site (Figure 5.11 & 5.12). From the home page, the subjects could navigate through the Bus site, which consisted of 157,637 web pages.

Figure 5.10 shows a part of the compositional structure of the web-enabled TV set. All button clicks were received by the Router, which passed it on to the Television or the Browser interaction component, depending on which one was selected at that moment. The function of the Browser was to display a web page and make it possible to select a link in the web page. When the subjects activated a link, or requested the home or previous page, the Browser sent this request to the Web Pages interaction component. The Web Pages interaction component retrieved the required web page and passed it back to be displayed.

The experiment had a 2 (web pages) \times 2 (browser) design. Variation in the web page's



Figure 5.11: Matrix page layout with links both on the same line and one below the other.



Figure 5.12: List page layout with links only one below the other.

layout led to two versions of the Web Pages interaction component. One layout, the *matrix* layout, placed the web links in a web page both on the same line and one below the other (Figure 5.11). The other layout, the *list* layout, placed all links one below the other (Figure 5.12). On the PC screen, the image of the remote control was displayed in an upright position to the right of the image of the TV set. Variations in the remote control led to two versions of the Browser interaction component. For one remote control, the *linear*-oriented version (Figure 5.13 right), the Up and Down buttons were interpreted as, *select the previous link* or *select the next link in succession*. The sequence went from left to right and continued on the left of the next line. The Left and Right buttons were interpreted as *jumping to the previous web page* and *activate the selected link*. For the other remote control, the *plane*-oriented version (Figure 5.13 left), the Up and Down buttons were interpreted as *select the link above* and *select the link below the current link selected*. Consequently, the Left and Right buttons were interpreted as *select the link left* or *right from the current link selected*. The subjects could jump to the previous web page with the Back button and activate the selected link with the Middle button.

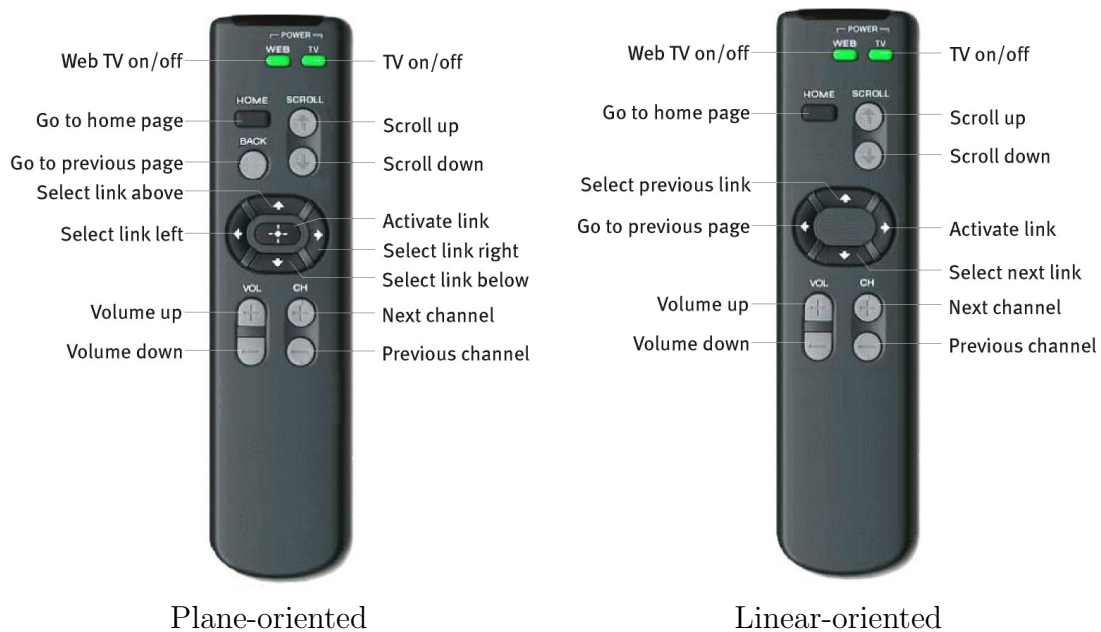


Figure 5.13: The remote controls and their functions, which subjects could use in the experiment. Left, the plane-oriented remote control with the Left and Right arrows to select the left and right links. Right, the linear-oriented remote control with the Left arrow button to jump to previous web page and the Right arrow button to activate selected link.

Combining the different versions led to four prototypes. Of the four prototypes, the prototype combining the matrix page layout and the linear-oriented browser was expected to result in the lowest performance, ease-of-use and satisfaction of the Browser interaction component. This low usability would not be explainable by the main effects of Browser

Table 5.5: Results of multivariate and univariate analyses of variance on the performance measures of the web-enabled TV set for the independent variables Browser version and Web Pages version.

Measure	Browser version			Web Pages version			Browser version \times Web Pages version		
	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>
Joint measure	3, 42	11.38	<0.001	3, 42	9.21	<0.001	3, 42	10.13	<0.001
Keystrokes	1, 44	15.21	<0.001	1, 44	3.31	0.076	1, 44	7.89	0.007
Messages Browser	1, 44	13.78	0.001	1, 44	3.32	0.075	1, 44	6.79	0.012
Messages Web Pages	1, 44	24.22	<0.001	1, 44	15.62	<0.001	1, 44	16.82	<0.001

and the Web Pages versions separately, but was expected as a result of an interaction effect between the versions of the two interaction components.

5.4.2 Results

Performance

The minimal number of keystrokes and messages received by the Browser required to perform the tasks were different in the four prototypes. Therefore, instead of analysing these absolute numbers, the number of keystrokes and messages received by the Browser that were needed in addition to the minimal numbers were analysed by subtracting the minimal numbers from the observed ones. Because the task time could not be corrected in this way, analyses of the task time were not carried out.

A MANOVA was conducted on the keystrokes, the number of messages received by the Browser and the number of messages received by the Web Pages interaction component. The analysis took the versions of the Browser (linear- or plane-oriented) and the Web Pages (matrix or list layout) as between-subjects variables. The result of the analysis is presented in Table 5.5.

The multivariate analysis (joint measure) found significant main effects for the Browser version and the Web Pages version. Furthermore, the analyses also found a significant two-way interaction effect between the Browser version and the Web Pages version. The individual measures were subjected to individual univariate analyses with the same between-subjects variables as in the multivariate analysis. The results are also presented in Table 5.5.

The analysis of the number of keystrokes found a significant main effect for the Browser version. The subjects made more keystrokes when operating the linear-oriented than the

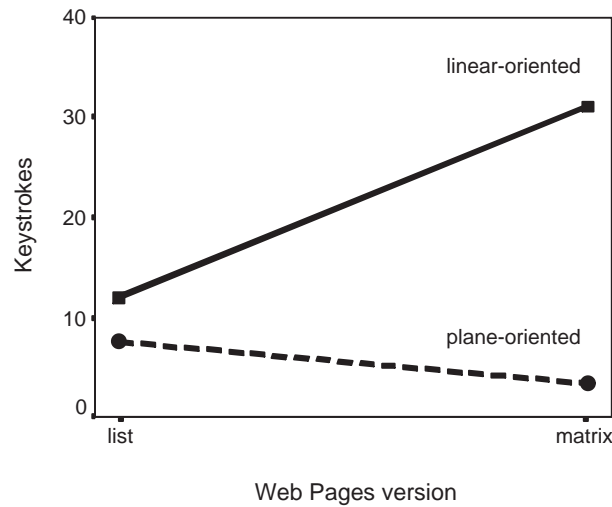


Figure 5.14: Number of keystrokes (needed in addition to minimal number) the subjects made when operating the web-enabled TV set with the two different browser versions.

plane-oriented browser (Figure 5.14). The analysis also found a significant two-way interaction effect between the Browser version and the Web Pages version. The explanation for this effect is that more keystrokes were made in the prototype that combined the linear-oriented browser and the matrix Web Pages than in the other prototypes (Figure 5.14).

The analysis of the number of messages received by the Browser interaction component revealed practically the same result as the analysis on the keystrokes. The analysis also found a significant main effect for the Browser versions. More messages were received when prototypes were equipped with the linear-oriented browser than with the plane-oriented version (Figure 5.15). The analysis also found a significant two-way interaction effect between the Browser version and the Web Pages version. The explanation for this effect is that more messages were received in the prototype that combined the linear-oriented Browser and matrix Web Pages versions than in the other three prototypes (Figure 5.15).

The analysis conducted on the number of messages received by the Web Pages interaction component revealed significant main effects for the Browser version and Web Pages versions. The Web Pages interaction component received more messages when a prototype was equipped with the linear-oriented instead of the plane-oriented version of the Browser, and when a prototype was equipped with the matrix instead of the list version of the Web Pages interaction component (Figure 5.16). The analysis also found a significant interaction effect between the Browser version and the Web Pages version. As before, the explanation was that more messages were received in the prototype that combined the linear-oriented Browser version and the matrix Web Pages version than in the other prototypes (Figure 5.16).

The standardised reception coefficient (SRC) of the browser was calculated to see whether

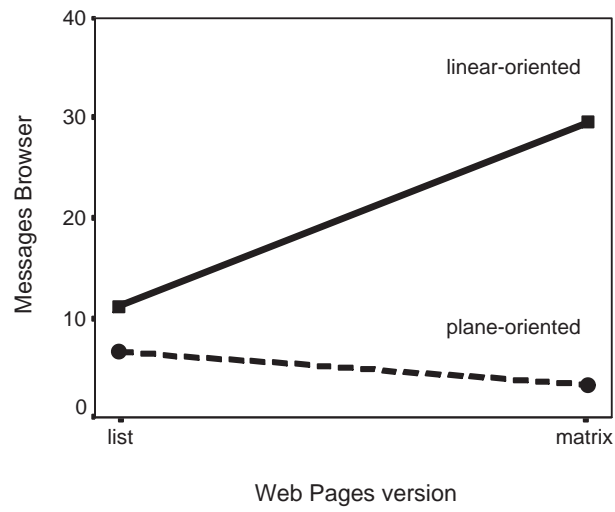


Figure 5.15: Number of messages received (needed in addition to minimal number) by the two Browser interaction component versions.

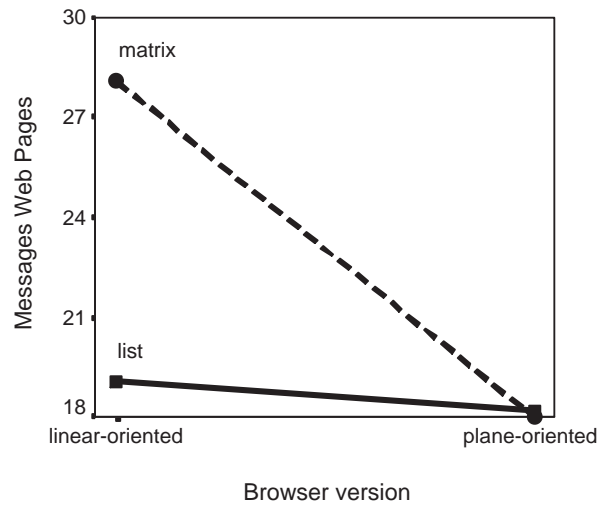


Figure 5.16: Number of messages received by the two Web Pages interaction component versions. At least 18 messages were needed to perform all tasks.

Table 5.6: Mean standardised reception coefficient values of the Browser interaction component.

Browser	Web Pages	
	List	Matrix
Linear-oriented	0.95**	1.18**
Plane-oriented	0.94**	0.94*

Note. H_0 : value = 1.

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

Table 5.7: Cronbach alpha of the questionnaire items related to the web-enabled TV set.

Measure	Description in questions	Coefficient Alpha	
		Ease of use	Satisfaction
Overall	The bus Internet site when accessed through the webtv	0.91	0.92
Browser	Webtv browser	0.90	0.90
Web Pages	Bus Internet site	0.89	0.86

the users sent unintentionally more messages to the Web Pages interaction component than required. The SRC is a ratio between the number of messages actually received and the number of messages the interaction component would receive if it operated efficiently while satisfying its higher-level demand (Equation 3.1).

The results (Table 5.6) show that the Browser's SRC value was only higher than 1 (optimal performance) when the prototype was equipped with the linear-oriented Browser version and matrix Web Pages version. In the three other prototypes, the browser's SRC value was below 1.

Questionnaire

Table 5.7 shows the results of the reliability analyses performed on the answers of the ease-of-use and satisfaction questions related to the web television experiment. All measures have an acceptable reliability level of more than 0.8. Again, for the ease of analysis, the averages of the answers on the ease-of-use and satisfaction questions were calculated for the analyses of variance. This reduced the 24 questionnaire answers for this experiment to 6 subjective measures.

Table 5.8: Results of multivariate and univariate analyses of variance on the ease-of-use measures of the web-enabled TV set for the independent variables Browser version and Web Pages version.

Measure	Browser version			Web Pages version			Browser version \times Web Pages version		
	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>
Joint measure	3, 42	3.58	0.022	3, 42	1.68	0.187	3, 42	1.96	0.135
Overall	1, 44	10.07	0.003	1, 44	3.38	0.073	1, 44	5.88	0.019
Webtv browser	1, 44	6.42	0.015	1, 44	4.13	0.048	1, 44	5.87	0.020
Bus Internet site	1, 44	7.24	0.010	1, 44	5.05	0.030	1, 44	5.63	0.022

Ease of use A MANOVA was conducted on the ease-of-use measure of the overall set-up (the bus Internet site when accessed through the webtv), the Browser and the bus Internet site. The analysis took the Browser version (2) and the Web Pages version (2) as between-subjects variables. The results are presented in Table 5.8.

The multivariate analysis (joint measure) only revealed a significant main effect for the Browser versions. The results of the univariate analyses of the individual ease-of-use measures are presented in Table 5.8 as well. The analysis of the overall ease-of-use measure (Bus site accessed through the webtv browser) found a significant main effect for the Browser version. The ease-of-use of the bus site when accessed through the webtv browser was rated higher when the prototype was equipped with the plane-oriented browser version than with the linear-oriented version (Figure 5.17). The analysis of the overall measure also found a significant two-way interaction effect between the Browser version and the Web Pages version. The explanation for this effect is the lower rating for a prototype that combined the linear-oriented Browser version and the matrix Web Pages version than for the other prototypes (Figure 5.17).

The analysis of the ease-of-use rating of the browser revealed significant main effects for the Browser version and the Web Pages version. The ease-of-use of the browser was rated lower when a prototype was equipped with the linear-oriented instead of the plane-oriented browser version, or with the matrix instead of the list version of the Web Pages interaction component (Figure 5.18). Furthermore, the analyses revealed a two-way interaction effect between the Browser version and the Web Pages version, which can be explained by the lower browser rating in the prototype equipped with the linear-oriented Browser version and the matrix Web Pages version than in the other prototypes (Figure 5.18).

The univariate analysis of the ease-of-use rating of the bus Internet site revealed main effects for the Browser version and for the Web Pages version. The ease-of-use of the bus Internet site is rated lower when the prototype was equipped with the linear-oriented instead of the

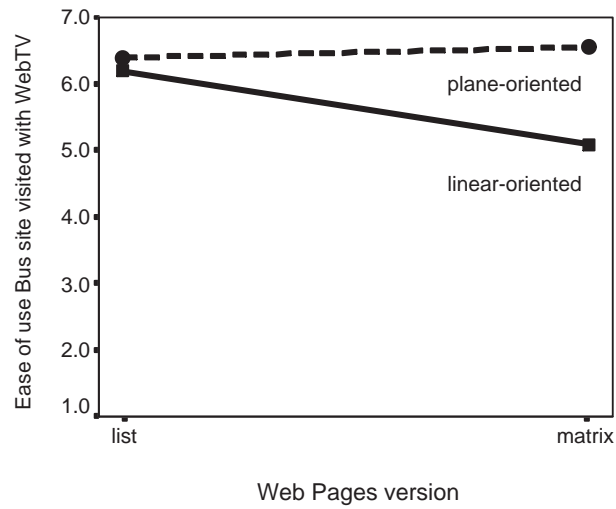


Figure 5.17: Ease-of-use ratings of the *bus Internet site when accessed through the webtv* for the two browser versions.

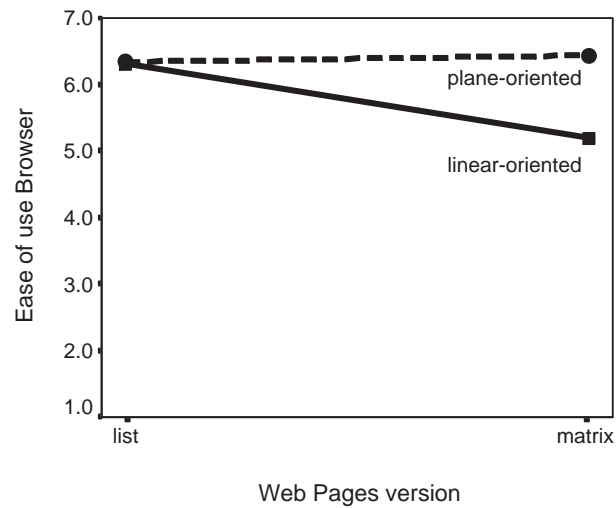


Figure 5.18: Ease-of-use ratings of the *webtv browser* for the two browser versions.

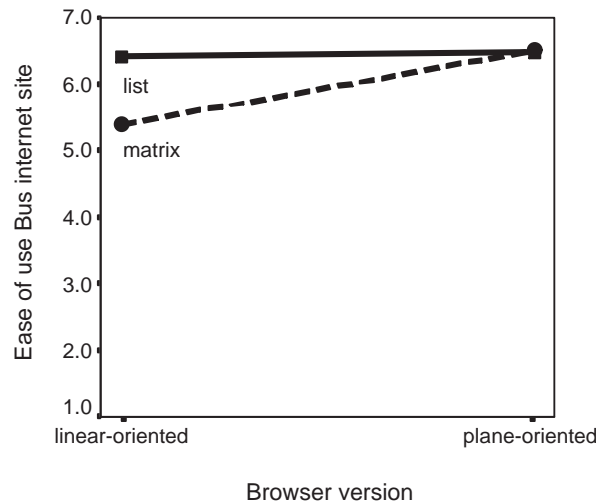


Figure 5.19: Ease-of-use ratings of the *bus Internet site* for the two Web Pages versions.

plane-oriented browser, or when the pages were presented in a matrix layout instead of the list layout (Figure 5.19). The analysis also found a significant two-way interaction effect between the Browser version and Web Pages version. Again, the explanation is the lower rating of the bus Internet site for the prototype that combined the linear-oriented browser and the matrix Web Pages version compared to the other prototypes.

Satisfaction A MANOVA was conducted on the satisfaction rating of the bus Internet site when accessed through the webtv, the Browser, and the Bus Internet site. The between-subjects variables were the same as in the previous analyses. Table 5.9 shows the results of both the multivariate and univariate analyses on the individual satisfaction measures.

The multivariate analysis (joint measure) revealed a significant main effect for the Browser version. Furthermore, the analysis found a significant two-way interaction effect between the Browser version and the Web Pages version. Again, the univariate analyses gave insight into the direction of the effects. The analysis of the overall satisfaction measure for the *Bus Internet site when accessed through the webtv browser* found a significant main effect for the Browser version. The overall satisfaction was rated lower for a prototype equipped with linear-oriented version than with a plane-oriented version (Figure 5.20). The analysis also revealed a two-way interaction effect between the Browser version and the Web Pages version. The reason for this effect can be seen in Figure 5.20. The subjects rated the prototype combining the linear-oriented Browser version and matrix version of Web Pages lower than all the other prototypes.

The analysis of the satisfaction of the webtv browser revealed significant main effects for the Browser version and the Web Pages version. Figure 5.21 shows that the subjects rated the satisfaction higher for prototypes equipped with the plane-oriented, instead of the

Table 5.9: Results of multivariate and univariate analyses of variance on the satisfaction measures of the web-enabled TV set for the independent variables Browser version and Web Pages version.

Measure	Browser version			Web Pages version			Browser version \times Web Pages version		
	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>
Joint measure	3, 42	4.85	0.005	3, 42	2.68	0.059	3, 42	4.92	0.005
Overall	1, 44	11.87	0.001	1, 44	3.30	0.076	1, 44	12.75	0.001
Webtv browser	1, 44	15.21	<0.001	1, 44	5.71	0.021	1, 44	15.21	<0.001
Bus internet site	1, 44	8.58	0.005	1, 44	7.82	0.008	1, 44	10.21	0.003

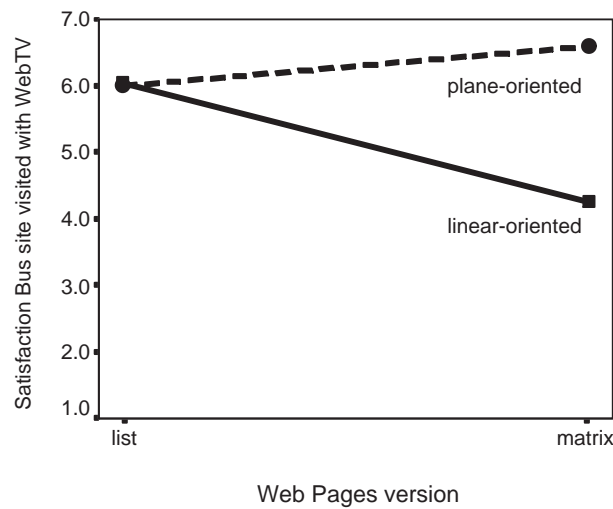


Figure 5.20: Satisfaction ratings of the *bus Internet site when accessed through the webtv* for the two browser versions.

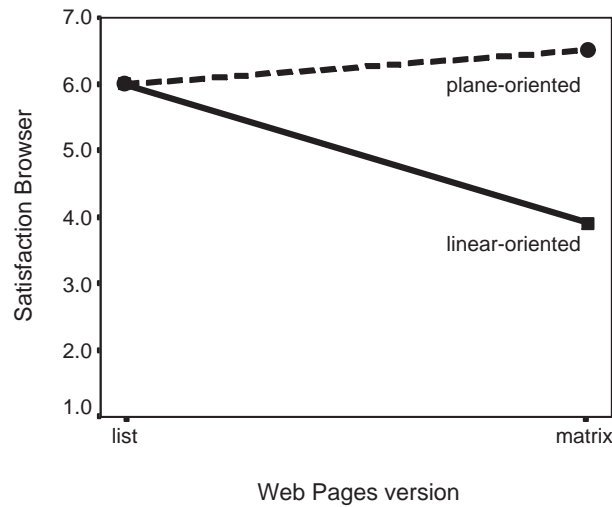


Figure 5.21: Satisfaction ratings of the *webtv browser* for the two browser versions.

linear-oriented browser version, and with the list instead of the matrix page layout. The analysis also found a significant two-way interaction effect between the Browser version and the Web Pages versions. The explanation is the lower browser rating subjects gave to the prototype combining the linear-oriented browser and matrix page layout compared to the other prototypes (Figure 5.21).

The analysis of the satisfaction rating of the bus Internet site revealed significant main effects for the Browser version and the Web Pages version. Figure 5.22 shows that subjects rated satisfaction of the web site higher when a prototype was equipped with the plane-oriented instead of the linear Browser version, and with the list instead of the matrix page layout. In addition, the analysis found a significant two-way interaction effect. An explanation for this interaction effect can be found in Figure 5.22. The subjects rated the bus Internet site lower for the prototype combining the linear-oriented browser version and the matrix page layout compared to the other prototypes.

5.4.3 Discussion

The interaction effects found between the Browser version and the Web Pages version in the number of keystrokes, in the number of messages received by the Browser and Web Pages, in the ease-of-use and in the satisfaction rating of the *Bus site accessed through the webtv browser*, the browser and the bus Internet site, all support the initially stated hypothesis. The prototype combining the linear-oriented browser and matrix web pages layout was correctly hypothesised to have a lower usability than the other prototypes. The experiment shows that inconsistency between interaction components that operate on different layers can affect the usability of the individual interaction components. The

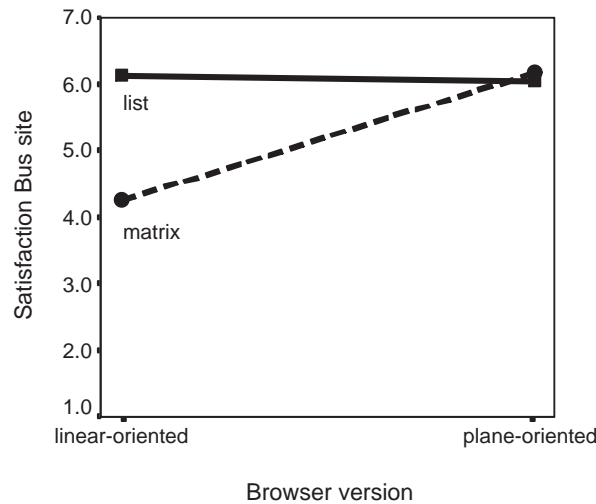


Figure 5.22: Satisfaction ratings of the *bus Internet site* for the two Web Pages versions.

usability reduction of the lower-level interaction component can be explained by the application of an inappropriate mental model. However, the explanation for the usability reduction of the higher-level interaction component is not clear yet. One explanation is that this was caused by an effectiveness reduction of the lower-level interaction component. When operating the prototype, in which the matrix and the linear-oriented browser versions were implemented, the subjects more often requested the retrieval of web pages than in other prototypes. Presumably, the subjects unintentionally activated links because they mistook the Right button for selecting the right link, instead of activating the current link. This means that the subjects sent more messages to the higher-level interaction component than they wanted, which is defined as an effectiveness problem of the lower-level interaction component (chapter 3). The results of the browsers' SRC—an ineffectiveness indicator—confirmed this idea. The SRC value was significantly higher than one, indicating ineffectiveness, for the prototype with the linear-oriented Browser version and matrix Web Pages version. This means that although the higher-level interaction component partly caused the activation of an inappropriate mental model to be applied to the lower-level interaction component, the higher-level performance reduction can not solely be attributed to the usability of the higher-level interaction component. However, this does not explain why the subjects rated the ease-of-use of the Web Pages, i. e. the bus Internet site, lower and were less satisfied with it when they had operated this prototype instead of the other three prototypes.

Three explanations can be offered for the interaction effect in the perceived ease-of-use and satisfaction of the Web Pages: (1) the subjects made no distinction between the two interaction components; (2) the subjects rated the Web Pages interaction component for suitability in the context of the Browser; (3) or the subjects made no distinction in their

control experience of the two interaction components. The first explanation suggests that the subjects could not separate the browser from the bus site. However, this is unlikely. The subjects were reasonably experienced Internet users, especially as most of them have a laptop with an Internet connection as part of a university program. The second explanation suggests that the subjects ignored the object of evaluation and answered the questions as they were supposed to answer the overall questions, which referred to the *bus Internet site when accessed through the webtv*. This is not in line with the previous study described in chapter 4, where the subjects were able to evaluate interaction components. However, the previous study also suggested that the subjects took into account their control experience of other interaction components when rating an interaction component's ease-of-use. This fits well with the third explanation. The subjects did not separate their control experience. The control experience of the browser could have affected the subjects' emotional state. Their emotional state in turn could have affected their control experience of the bus site and consequently their bus site evaluation. Which is not unlikely since the emotional state, e. g. mood, has proven to affect people when they evaluate products (Curren & Harich, 1994).

5.5 Experiment 3 —consistency between the component and the application domain

Does the usability of a data browser change when it is deployed in a mobile telephone to navigate through text messages or in a digital camera to navigate through pictures? The last experiment studied the effect the application domain has on the usability of an interaction component. The application domain may activate a general mental model, which in turn may activate a component-specific mental model, which users apply to control an interaction component (Figure 5.1). The difference with the previous experiments is that it is not the E-feedback of other components, but the users' idea of operating a particular device that determines what component-specific mental model they apply. When a general mental model affects the usability of an interaction component, user interface designers should apply an integral metaphor that suits the application domain, instead of individual metaphors for each component, or metaphors that do not fit with the application domain.

The experiment took a radio alarm clock and a microwave as applications in which two versions of a clock were implemented. In the radio alarm clock (Figure 5.23), the clock determined when the radio should be switched on, and in the microwave (Figure 5.24), the clock determined when the cooking should start.

5.5.1 Method

The task the subjects had to perform with the microwave was to set a timer, the cooking time and the power. For the radio alarm clock, the subjects had to set the alarm time, the



Figure 5.23: User interface of the radio alarm clock with a hot dish presented next to the timer time.



Figure 5.24: User interface of the microwave with a ringing mechanical alarm clock presented next to the timer time.

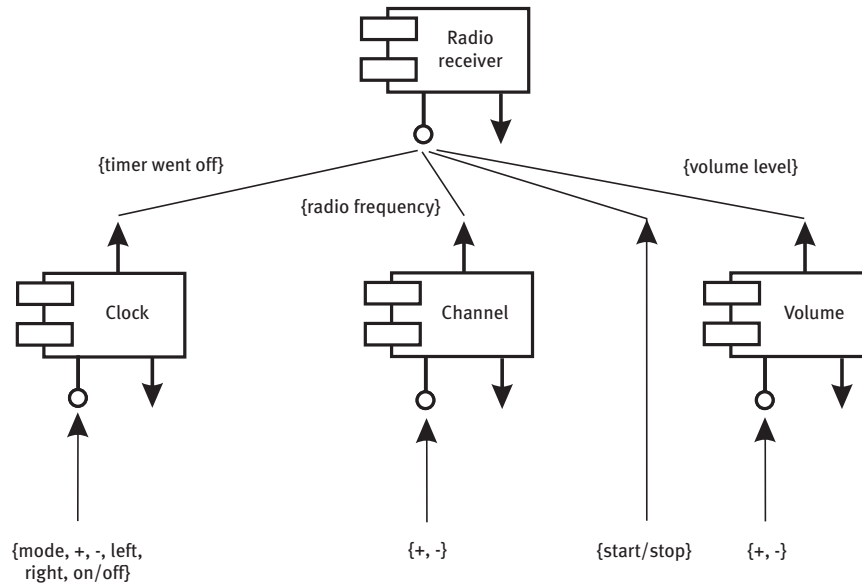


Figure 5.25: Part of the compositional structure of the radio alarm clock.

radio channel and the volume. Both the compositional structures of the radio alarm clock (Figure 5.25) and the microwave (Figure 5.26) included the interaction component Clock. This interaction component was responsible for the current time and the timer time. To see and to set these times, the subjects had to press the Mode button to put the clock in the required mode. After this, the subjects could press the $+$ and $-$ button to increase or decrease a digit and the Left and Right button to select another digit. The subjects activated the timer with the *On/Off* button. When the timer went off, the message *time went off* was sent to the Period interaction component in the case of the microwave or to the Radio Receiver interaction component in the case of the radio alarm clock.

The fit or misfit between the application domain and the clock was in the clock's E-feedback that was presented along with the timer time. In one version, the mechanical alarm version, the symbol of a ringing mechanical alarm clock was shown, in the other version, the hot dish version, a symbol of a hot dish (Figure 5.27). The clock had four different modes: displaying the current time, displaying the timer time, setting the current time, and setting the timer time. The current time was presented along with a symbol of a clock (Figure 5.27). The timer time was presented along with the ringing mechanical alarm clock or the hot dish.

When the subjects performed a task with the radio alarm clock, the expectation was that the task of setting the alarm of an alarm clock would activate a general model on alarm clocks, which subsequently activates a component-specific mental model on setting the alarm of alarm clocks. In light of this activated component-specific mental model, subjects could more easily understand the E-feedback *the time the timer will go off* or in this domain *the time that the alarm will go off* by being presented with the ringing mechanical

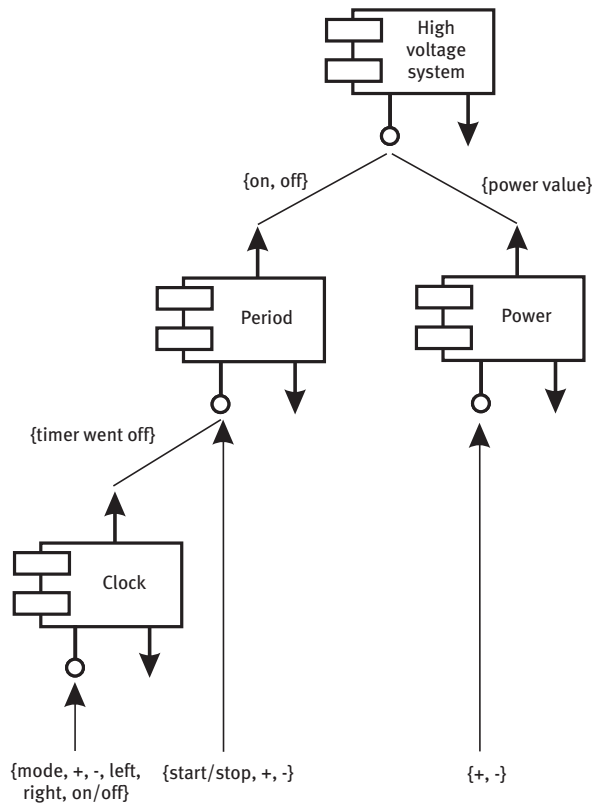


Figure 5.26: Part of the compositional structure of the microwave.

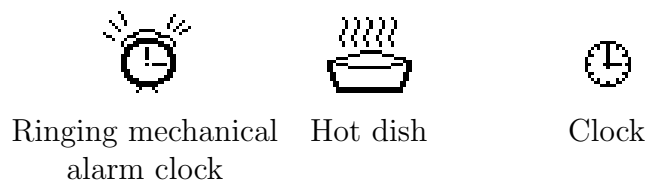


Figure 5.27: Symbols presented along with the current time and the timer time. On the left and in the middle, symbols that were presented with the timer time, (left) a ringing mechanical alarm clock, and (middle) a hot dish. On the right, the clock symbol that was presented along with the current time.

Table 5.10: Results of multivariate and univariate analyses of variance on the performance measures of the radio alarm or the microwave for the independent variables Clock version and Application domain.

Measure	Clock version			Application domain			Clock version × Application domain		
	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>
Joint measure	2, 43	0.59	0.560	2, 43	3.73	0.032	2, 43	0.01	0.988
Keystrokes	1, 44	0.87	0.356	1, 44	4.05	0.050	1, 44	0.02	0.885
Mode messages	1, 44	1.19	0.282	1, 44	7.57	0.009	1, 44	0.02	0.877

alarm clock symbol than being presented with a hot dish symbol. In terms of semiotics, the subjects could more easily associate the signifier with the signified (Chandler, 2002).

The opposite was expected for subjects that perform a task with the microwave. Setting the microwave was expected to activate a general model on cooking devices, which subsequently activated a component-specific mental model on controlling cooking processes. In this domain, the same E-feedback of the clock would now signify *the time the cooking begins*. With the component-specific mental model activated, the hot dish symbol was expected to present this better than the ringing mechanical alarm clock symbol. The hypothesis was that the performance, perceived ease-of-use and satisfaction would be higher for prototypes where the symbol presented next to the timer time suited the application domain than for prototypes where this did not fit.

5.5.2 Results

Performance

As in the previous experiment, the minimal number of keystrokes required to perform the tasks were different in the prototypes. Therefore, instead of the absolute number, the number of keystrokes made in addition to the minimal number was analysed by subtracting the minimal number from the observed one.

A MANOVA was conducted on the keystrokes, and the number of Mode messages received by the Clock interaction component. The analysis took the Clock version (2) and Application domain (2) as between-subjects variables. Table 5.10 presents the results of both the multivariate and univariate analysis on these two measures. The multivariate analysis found a significant main effect for the Application domain. A significant main effect for the Application domain was also found in the univariate analysis of the keystrokes. The subjects needed more keystrokes for the microwave prototype than for the radio alarm clock (Figure 5.28).

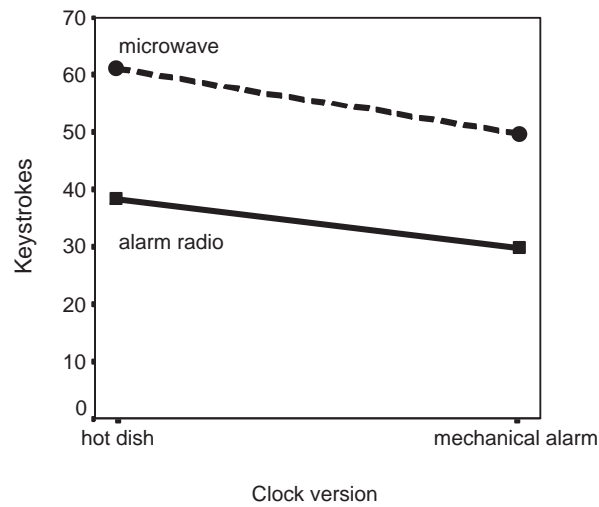


Figure 5.28: Number of keystrokes subjects made (in addition to the minimal number required) when operating the radio alarm clock or the microwave.

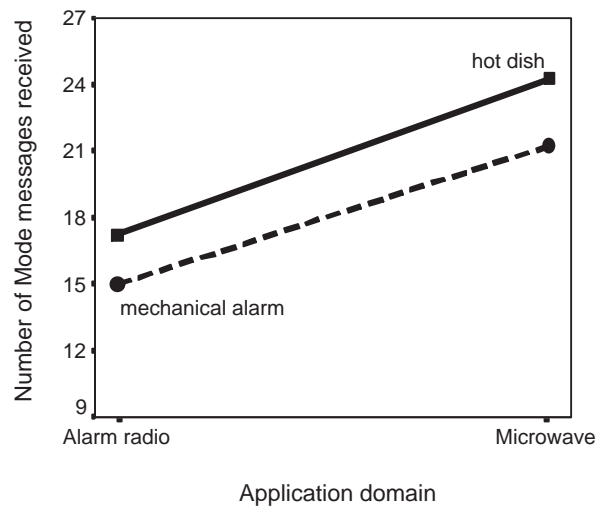


Figure 5.29: Number of Mode messages the two versions of the Clock interaction component received. At least 9 were required to perform all tasks.

Table 5.11: Cronbach alpha of the questionnaire items related to the radio alarm clock or the microwave.

Measure	Description in questions	Coefficient Alpha	
		Ease of use	Satisfaction
Application	radio alarm or microwave	0.94	0.84
Clock	clock/alarm or clock/timer	0.92	0.92

The univariate analysis of the number of Mode messages received by the Clock interaction component also found a significant main effect for the application domain. The subjects more often changed the clock mode when they operated the microwave than when they operated the radio alarm clock (Figure 5.29). Both the multivariate and univariate analyses did not find a significant main effect for the Clock version, or a significant two-way interaction effect between Clock version and the Application domain.

Questionnaire

The results of reliability analyses conducted on the answers of the ease-of-use and the satisfaction questions are given in Table 5.11. All measures have an acceptable reliability level of more than 0.8. As before, the averages of the 16 answers on the ease-of-use and satisfaction of the application and the Clock interaction component was calculated to create 4 subjective measures.

Ease of use A MANOVA on the ease-of-use measures of the applications and the clock with Clock version (2) and Application domain as between-subjects variables produced no significant effects (Table 5.12). However, the univariate analyses on the ease-of-use of the application and on the clock, revealed a significant effect for the Clock versions. The subjects felt the application as well as the clock was harder to use when the clock was implemented with the hot dish than with the ringing mechanical alarm symbol (Figure 5.30 and 5.31).

Satisfaction A MANOVA on the satisfaction measure of the application and the clock, with the same between-subjects variables, produced a significant effect for the Application domain (Table 5.13). This effect was also found in the satisfaction rating of the clock. Figure 5.33 shows that the subjects were more satisfied with the clock in the radio alarm than in the microwave. The univariate analysis of the satisfaction of the application revealed a significant main effect for the Clock version. The satisfaction of the application was higher

Table 5.12: Results of multivariate and univariate analyses of variance on the ease-of-use measures of the radio alarm or the microwave for the independent variables Clock version and Application domain.

Measure	Clock version			Application domain			Clock version \times Application domain		
	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>
Joint measure	2, 43	3.10	0.056	2, 43	2.28	0.114	2, 43	0.78	0.465
Application	1, 44	6.01	0.018	1, 44	0.64	0.427	1, 44	1.19	0.281
Clock	1, 44	6.25	0.016	1, 44	2.02	0.162	1, 44	0.67	0.417

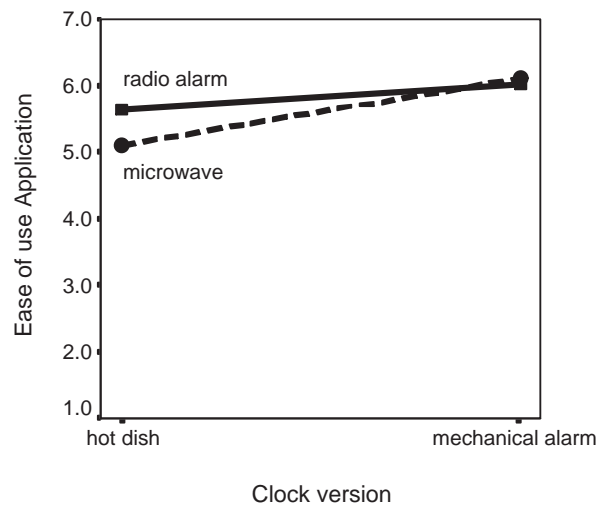


Figure 5.30: Ease-of-use rating of the radio alarm clock and the microwave.

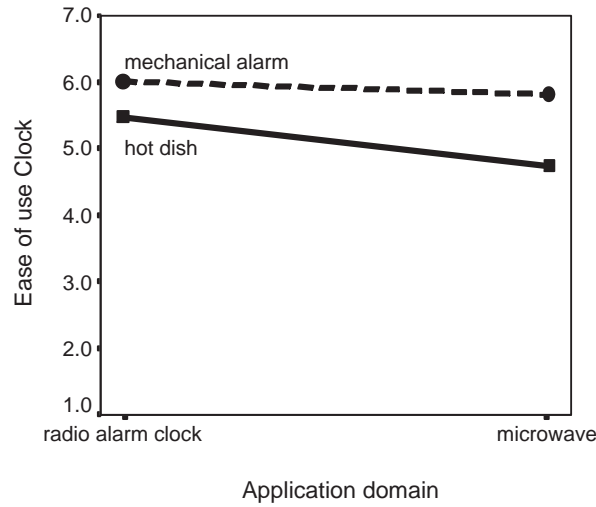


Figure 5.31: Ease-of-use rating of the two Clock versions.

when the application was equipped with the ringing mechanical alarm Clock version than with the hot dish version (Figure 5.32).

5.5.3 Discussion

The lack of significant interaction effects between the Clock version and the Application domain in the analysis of the performance, the perceived ease-of-use and the satisfaction measures does not offer support for the hypothesis that inconsistency between the application domain and the clock reduces the clock's usability. Besides the straightforward interpretation that there is no general mental model that indirectly influences the control

Table 5.13: Results of multivariate and univariate analyses of variance on the satisfaction measures of the radio alarm or the microwave for the independent variables Clock version and Application domain.

Measure	Clock version			Application domain			Clock version × Application domain		
	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>	<i>df</i> 's	<i>F</i>	<i>p</i>
Joint measure	2, 43	2.02	0.146	2, 43	7.92	0.001	2, 43	0.54	0.589
Application	1, 44	4.13	0.048	1, 44	2.02	0.162	1, 44	0.01	0.920
Clock	1, 44	3.34	0.075	1, 44	8.48	0.006	1, 44	0.30	0.590

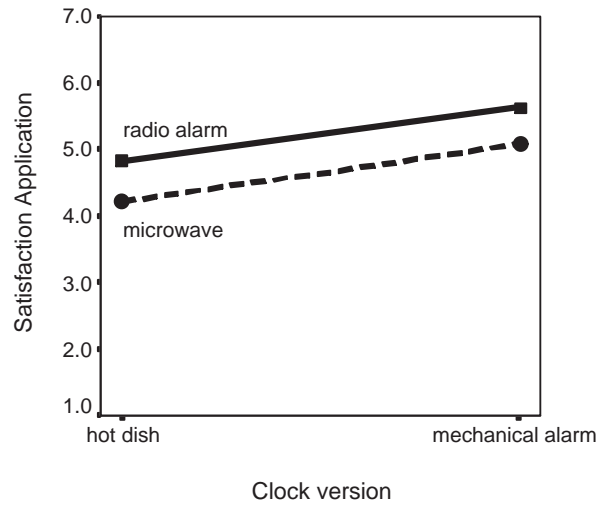


Figure 5.32: Satisfaction rating of the radio alarm clock and the microwave.

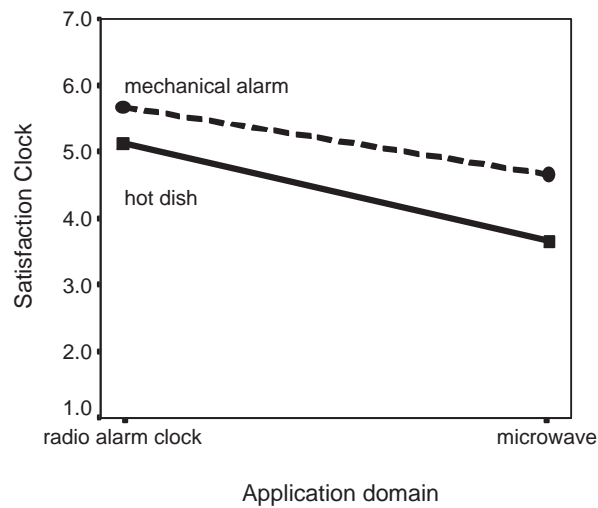


Figure 5.33: Satisfaction rating of the two Clock versions.

of a specific interaction component, another interpretation is an unanticipated effect of the experimental set-up. Although the subjects may not have understood the inconsistent symbols presented along with the timer time, the other E-feedback of the application did not suggest looking elsewhere. Consequently, the subjects were just left with the only conclusion that this inconsistent symbol had something to do with the timer time. However, this may explain the result of the analyses on the performance measures, but not why no interaction effect was found in the ease-of-use or satisfaction measures other than that the subjects based their rating on the performance. However, this latter idea is not in line with the findings. The performance measures only revealed an effect for the application domain whereas the ease-of-use rating of the application and the clock, and the satisfaction rating of the application revealed an effect for the Clock versions. The subjects gave a lower rating when a prototype was equipped with the hot dish symbol instead of the ringing mechanical alarm clock. The ringing mechanical alarm clock was probably a more common presentation of the timer time, or it fitted better with the current time symbol, which was a clock (Figure 5.27).

The finding that subjects needed less keystrokes, when operating the radio alarm clock instead of the microwave, indicated that the usability of the radio alarm clock was higher than the microwave. The findings that the subjects less often changed the clock mode, and were more satisfied with the clock when they operated the radio alarm than when they operated the microwave is more interesting. The same clock function was clearer in one application domain than in the other. This means that the usability of the interaction component depends on the application domain, which undermines LPT's claim that components operate independently. Two explanations can be given for this finding. First, the subjects could understand the concept of a timer better in relation to the alarm clock since it is more related to the main function of the alarm clock, which is waking someone, than to the main function of the microwave which is preparing a meal. Second, the other components of the microwave had a negative impact on the usability of the clock. When operating the microwave, the subjects may have had a problem distinguishing the Clock and the Period interaction component (Figure 5.26), which also deals with time—the cooking time.

However, this finding does not indicate that a particular interaction component *version* fits in better with particular application domains than other versions do, which the experiment was expected to show.

5.6 General discussion

5.6.1 Theoretical implications

The results of the first two experiments show that the control of interaction components can depend on other interaction components. E-feedback is interpreted with a component-specific mental model, which E-feedback of other interaction components may have acti-

vated. The third experiment shows that the application domain may also have an impact on the usability of interaction components. Here it was not inconsistency between the E-feedback and the application domain, but that the functionality an interaction component offers may be clearer in one application domain than in the other.

These findings may be limited to the phase where users learn to control an interaction component as was the case in all three experiments. Once users gain experience with controlling interaction components the dependency between them may lessen because the correct component-specific mental model will be active. In the initial phase, users are guided by E-feedback and later on they might be more directed by their own experience.

The dependency between components does not confirm LPT's claim. The theory states that the user-system interaction within a layer is independent of the other layers, to the degree that they only exchange messages. However, the theory limits itself if it suggests that control processes are solely based on the present interaction within one layer. Users rely on their knowledge gained from previous interaction. Although LPT may be right in stating that a control process aims at controlling a specific layer, it should not be concluded that it only relies on the interaction component's feedback to select actions.

5.6.2 Practical implications

The results of the experiment suggests that interaction component designers should be aware that the usability of their interaction component could be dependent on other interaction components to be deployed in a new user interface. First of all, designers should try to predict which other interaction components will be used in relation with their interaction component. If this is not possible, the interaction component could be designed according to a set of specific rules. Later on, in the reuse phase when the new user interface is constructed, the developers should make sure that the interaction components they apply follow the same rules, or at least that there are no conflicting rules. These rules can be laid down in a style guide. However, this does not guarantee a user interface without inconsistency, because users do not have to agree with what designers consider consistent. Only the involvement of users can solve this problem. Testers in a usability test who evaluate only one version of a user interface (the single version testing paradigm) should be aware that usability problems may not solely be attributed to one interaction component, but that the combination of two or more interaction components may cause them.

Once again, the component-base usability testing framework proved to be a useful evaluation method. In the first experiment, significant interaction effects were only found in the analysis of the component-specific measures. Both the objective and subjective component-specific measures were more powerful than their overall counterparts. The SRC indicator showed its use in the second experiment. The results of SRC analysis confirmed the explanation that a lower-level layer caused the interaction effects found in the higher-level interaction. Applying component-specific measures in the third experiment

revealed that the same interaction component could have a usability difference when implemented in different applications domains. This latter effect could only be found with a component-specific measure since a difference in the overall measures would simply be explained as a usability difference between the two applications.

5.7 Conclusions and further research

By using the component-base usability testing framework described in chapter 3, this study was able to show that interaction components in a user interface can affect each other's usability significantly. Interaction components in the same layer or in other layers can activate an inappropriate component-specific mental model, which users apply to understand the interaction component's feedback to select actions. The inconsistency between the application domain and an interaction component's E-feedback was not found to affect the interaction component's usability. Whether this only was the case in this experiment or can be generalised, is a topic for further research. However, the study did show that the application domain had an effect on the users' understanding of the functionality the component provides.

The following chapter describes an experiment about another kind of factor that could cause interaction components to lose their independence —mental effort.

Chapter 6

Effect of mental effort on the usability of user interface components

6.1 Introduction

In chapter 5, consistency was studied as a factor that makes that the usability of the entire user interface unpredictable based solely on the usability of the separate components. This chapter describes a study of another such factor: mental effort. When faced with a control strategy that is mentally too demanding to maintain, users may start deploying other strategies at the expense of efficient control to reach the primary goal or to remain within acceptable operational limits (e.g., Cnossen, 2000; Hockey, 1997; Meister, 1976; Sperandio, 1971). To make this more concrete, consider a driving example. After waiting for a traffic light to turn to green, student-drivers may turn a corner still driving in the first gear and only putting the car in the second gear after they have taken the corner. Although, the first gear is mainly intended to get the car moving after a standstill and not to drive in, student-drivers may be unable to change up fast enough before the corner while remaining in control of the car in the bend. To avoid a loss of control over the car, students end up taking the corner at a lower speed. In this case, improving the students' skill in changing the gear is more effective than improving their steering skills.

In this study, mental effort is proposed as a factor that links lower-level interaction component with the control strategy of high-level interaction components. The objective of the study reported here was to test this idea experimentally. Two calculators were designed with different versions of a lower-level interaction component. With these calculators, the mental effort and the higher-level control strategy was recorded when users calculated equations with varying difficulty.

6.2 Method

The experiment involved an application, written in Delphi 5, consisting of two calculator prototypes, a recording mechanism for the message exchange between the interaction components, task instructions, and questionnaires. The sections below describe the calculator prototypes, the hypothesis, the tasks the subjects had to perform, the measures that were deployed, the experimental design, the subjects who participated and the experimental procedure.

6.2.1 Prototypes

A calculator was chosen for this experiment because of its two-layered architecture. The lower-level component established an equation and the higher-level component processed it. Figure 6.1 shows the interaction structure. The two interaction components that were distinguished are the *Editor* and the *Processor & Memory interaction* components. The low-level interaction component, Editor, was responsible for forming the equation and passing it on to the higher-level interaction component Processor & Memory. The subjects could enter a complete equation, including special mathematical function (sine, ln, square root, etc), which would be passed on after the subjects pressed the '=' button, or the 'STO' button followed by a memory button (M1 to M6). The Editor only sent an equation upward if the number of opening and closing brackets in the equation were the same. If this was not the case, the Editor displayed an error message.

The high-level interaction component, Processor & Memory, processed the equation and placed the result in one of six memory places if requested. The equation was processed according to mathematical priority rules; calculation started with operations within the brackets, then multiplication and division, and finally addition and subtraction. When processing, the interaction component replaced the memory references in the equation (M1 to M6) with the actual values in the memory places. The result of the equation was sent back to the Editor; this could either be a value or an error message in the case of a division by zero or if an illegal value was fed into a special mathematical function.

Two versions of the Editor interaction component were designed: one with a large display (Figure 6.2), the other with a small display (Figure 6.3). The small display showed only a small part of an equation, for example: a value, an operator ('+'), a mathematical function name ('sin'), a reference to a memory place ('M1'), or a list of consecutive opening and closing brackets ('(((('). The large display showed the equation on a screen of five lines with 34 symbols each. If the equation was longer, the subjects could scroll up and down in the large display.

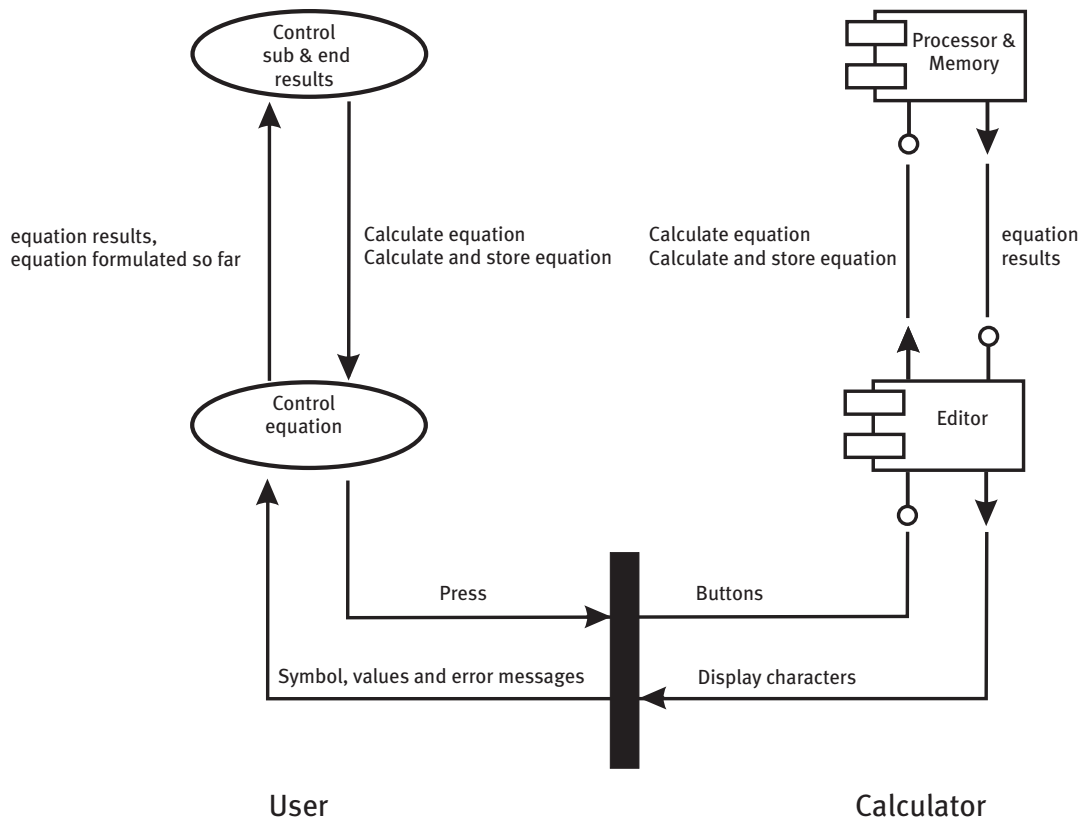


Figure 6.1: Layered interaction structure between a user and a calculator. The expectation feedback is omitted.

6.2.2 Hypothesis

In this experiment, the hypothesis was tested that mental effort creates a link between the Editor versions and the control strategy of the Processor & Memory interaction component, which contradicts LPTs' claim that the usability of a component is independent of other components in the user interface. The advantage of the large display over the small display was expected to be apparent when an equation was deeply nested. When subjects were entering an equation, as presented in Figure 6.4, they would have to keep track of the depth of the nesting and their location in the equation. With the large display, subjects could check this information from the input on the display (message 'equation formulated so far' in Figure 6.1), whereas in the case of the small display, subjects would have to memorise this. The depth of nesting was expected to relate to the difficulty of an equation. However, subjects might have the same control over the higher-level interaction component by spending more mental effort. Therefore, the general memory load of the task was increased in an attempt to exceed the effort subjects could invest, when they had to solve a deeply nested equation. The memory load was increased by two factors. First,

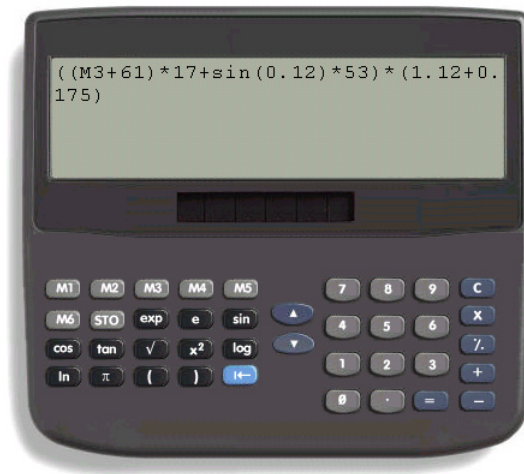


Figure 6.2: Calculator with large display.

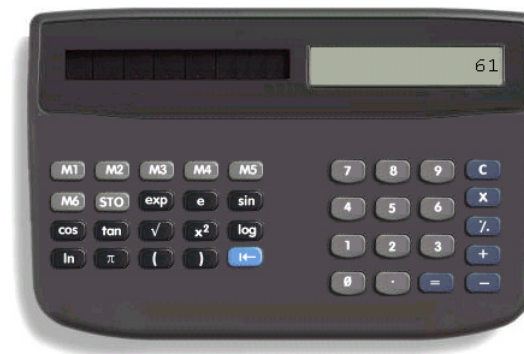


Figure 6.3: Calculator with small display.

subjects had to memorise the memory place of intermediate outcomes, which were stored in advance. Second, the subjects were interrupted to perform another task. They had to memorise where they left off, so that they could continue with the calculations after the interruption.

6.2.3 Alternative hypothesis

To conclude that mental effort is the underlying factor that caused the variation in the higher-level control, requires other factors to be ruled out, such as the ineffectiveness of the Editor. The experiments in chapter 4 and 5 illustrate that ineffectiveness problems of a lower-level interaction component can also influence the control of higher-level layers. In this experiment, a possible ineffectiveness problem could be that subjects make more

$$((M3+61)*17+\sin(0.12)*53)*(1.12+0.175)$$

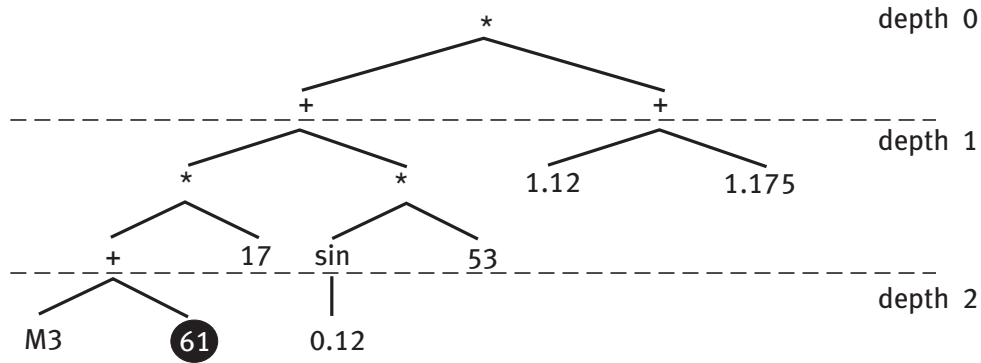


Figure 6.4: The same equation presented in an algebraic format and in a tree format. After entering the number 61, users have to realise that the equation requires at least two more closing brackets.

typing errors in the small display Editor than in the large display Editor, which results unintentionally in more messages to the Processor & Memory interaction component whose effects have to be undone. Therefore, an additional equation type was introduced, which required almost no mental effort to calculate. If the effect between the two versions of the Editor interaction component was caused by ineffectiveness, it should also become apparent when subjects solve this very easy equation. If not, the alternative hypothesis became very unlikely.

6.2.4 Tasks

Building project cost calculation

The intermediate outcomes, of which the subjects had to remember the memory places, were part of a building project cost calculation. The whole experiment was planned around a building company. The subjects were asked to calculate the cost based on a textual description of a building project (Figure 6.5). The seasonal labour cost per hour for the six different workers was randomly stored in the calculators' six memory places. The memory places and workers' labour cost were presented in a randomly ordered table on the PC screen for 40 seconds, before the calculator became visible and accessible. When calculating, subjects could use the intermediate outcomes stored. If they forgot the location of an intermediate outcome stored, they had to calculate the seasonal labour cost themselves, which always required the use of a sine or a cosine function. These two functions were only needed in the seasonal labour cost calculation and nowhere else. Therefore, the number of

times subjects entered these functions was an indicator of how well subjects remembered the location of the intermediate outcomes.

The subjects in the experiment had to solve equations of three levels of difficulty (difficult, easy and copy). Figure 6.5 illustrates a difficult equation. These equations had a nesting depth of two (Figure 6.6). The easy equations could be solved without applying brackets (nesting depth of zero). Instead of percentages, the absolute prices, including invoiced VAT, were given in the instruction text of these equations. The easiest equations were the copy equations, which were similar to the difficult equation, with the exception that with the instructions the solution was also presented in an algebraic format. Therefore, this equation was basically a copying task, which mainly involved the control of the Editor interaction component. The copy equation made it possible to study the potential effectiveness problem in lower-level interaction components as the alternative hypothesis. Although the equations had different levels of difficulty, all the equations required the same number of keystrokes if entered in an optimal form.

Interruption

A task that interrupted the cost calculation was included because interruptions have proved to negatively affect users' performance in mentally demanding tasks (e. g. Bailey, Konstan, & Carlis, 2001; Gillie & Broadbent, 1989; Kreifeldt & McCarthy, 1981). The computer interrupted the subjects 35 seconds after they started entering an equation. A telephone rang and a question was shown on the screen concerning the number of plumbers, carpenters, bricklayers, etc. that were planned in a particular week. The subjects had to look at a paper work schedule, search for projects where these workers were scheduled, and add them up. The solution always consisted of a summation of four numbers between 10 and 50. Only after the subjects selected the correct answer, did the application restore the previous screen, and could the subjects resume with the cost calculation task. During the interruption, the subjects could not see the calculator or the description of the building project. It was expected that the interruption was more memory demanding in the small display version because subjects had to recall where exactly they left off in entering the equation when the telephone rang. In the case of the large display, subjects could read this on the display. Subjects were not allowed to make any notes or to use a pen or pencil, during the whole experiment.

6.2.5 Measures

The component-based usability testing framework described in chapter 3 was used to measure the performance, the ease-of-use and satisfaction of the entire system, but also of the Editor and Processor & Memory interaction components separately. In accordance with this framework, the message exchange between the interaction components was recorded throughout the task execution. Objective overall performance measures obtain from this

Calculate the price for underfloor heating and house insulation

An amount of 3468 should be charged for materials and as a write-off of 1273 for the tools. A surcharge of 14.42% has to be paid for contingencies on labour, material and tools. This surcharge and an environmental surcharge of 106 are exempt from value-added tax (VAT). On all other costs (including labour cost) a 17.5% VAT is applied.

The renovation will take place in week number 47 ($S = 47\pi/52 = 2.84$), in which a reduction of 8.47% is applied on the hourly rate of the carpenter, and the electrician. Furthermore, the customer can claim a 12.37% reduction on the hourly rate of the fitter and the carpenter because of an advertising campaign of the Electricity Company. Extra workers have to be hired from other companies because of increased activity in the winter. Therefore, a surcharge of 3.78% has to be paid for the hourly rate of the painter and the fitter.

	Correction for season		Hourly rate	Hours
Painter	$14*\text{Sin}(S)$	→ M5	74	15
Bricklayer	$6*\text{Sin}(2*S)$	→ M2	72	12
Electrician	$6*\text{Sin}(3*S)$	→ M1	61	13
Foreman	$5*\text{Sin}(2*S)$	→ M6	85	11
Fitter	$18*\text{Cos}(S)$	→ M4	74	48
Carpenter	$12*\text{Cos}(S)$	→ M3	62	21

All prices are without VAT

Figure 6.5: Instruction text of a difficult equation. After 40 seconds the M-codes would disappear from the screen. In the case of a copy equation, letters that corresponded with the solution given would replace the M-codes.

$$\begin{aligned}
& \left(\begin{aligned}
& (M1 + 61) \times 13 \times (1 - 0.0847) + \\
& (M4 + 74) \times 48 \times (1.0378 - 0.1237) + \\
& (M2 + 72) \times 12 + \\
& (M6 + 85) \times 11 + \\
& (M5 + 74) \times 15 \times 1.0378 + \\
& (M3 + 62) \times 21 \times (1 - 0.0847 - 0.1237) + 3468 + 1273
\end{aligned} \right. \\
& \left. \right) \times (1.1442 + 0.175) + 106 =
\end{aligned}$$

Figure 6.6: Solution of Figure 6.5 in an algebraic format. In the copy equation task, this solution would be given in the task description. The M-codes would be replaced by letters, which would also replace the M-code after 40 seconds in the labour cost table.

recording were the task time and the number of keystrokes, which in this case was similar to the number of messages received by the Editor interaction component. The objective component-specific performance measures of the Processor & Memory interaction component were based on the number of messages this component received. This message stream was split up into two types of messages that were sent with the following requests: an equation to be processed including storing the result in one of its memory places (store-request); and an equation only to be processed (process-request). This led to two objective component-specific performance measures. An additional measure was obtained by looking inside these messages and count the number of times the execution of a mathematical function (sine or cosine) (function-request) was requested.

At the end of the experiment, the other subjective measures, ease-of-use and satisfaction (Appendix B), were obtained by administering a questionnaire about the ease-of-use and the satisfaction of the calculators, their Editor interaction component and their Processor & Memory interaction component. The questionnaire consisted of 48 questions that were presented in a random order. In the questionnaire, the calculator with a small display was referred to as the *small calculator* and the other calculator as the *large calculator*. The interaction components were referred to as the editor of the large calculator, the processor and memory of the large calculator, etc. Along with the questionnaire, a picture of the two calculators was given and a short definition of the Editor and Processor & Memory interaction component.

The mental effort invested was measured with a subjective and with a physiological measure. After the subjects solved an equation, they rated the effort on the Rating Scale Mental Effort (RSME) (Zijlstra, 1993). This one-dimensional scale employs verbally labelled anchor points that refer to an underlying continuum of effort expenditure. The scale was presented on the screen, after which the subjects could move an arrow to the appropriate position on the scale. Heart rate variability (HRV) was taken as a physiological measure. HRV relates to a person's mental effort (G.Mulder, 1980; L.J.M.Mulder, 1988) or, rather to the emotional reaction on the mental effort (Jorna, 1985). A person's HRV

is reduced during the performance of effort-full mental tasks. Especially the frequency band around 0.1 Hz, which relates to the baroreflex, regulating short-term blood pressure, has been found to be sensitive to mental effort. To record the electrocardiogram (ECG) signal, three silver-silver chloride (Ag-AgCl) lead electrodes were placed on the subjects, two on the chest and one on the back of the neck. The electrodes were connected to an electrocardiogram amplifier module (ECG100B). A Biopac MP100 data acquisition system converted the analogue signal into a digital signal, which was processed and captured with a sampling rate of 200 Hz by the AcqKnowledge 3.5.7. application running on a PC. The subjects were instructed not to speak, not to move their feet and legs too much, and not to touch the screen to avoid disturbances of the ECG signal.

6.2.6 Experimental design

The experiment had a 2×3 within-subjects design —editor (small or large) \times equation difficulty (difficult, easy, or copying). To control for learning and fatigue effects, a scheme was applied to 24 subjects, ordering the six conditions according to four different Latin squares that also counterbalanced for immediate sequential effects (Lewis, 1989).

6.2.7 Subjects

There were 28 subjects who participated, all students of Eindhoven University of Technology. They were expected to have extensive experience with calculators and be acquainted with mathematical priority rules. However, they were not expected to have experience with calculating the building cost in the way it was done in this experiment. In soliciting subjects for the experiment it was mentioned that because of the ECG measuring only students without heart problems could enter the experiment.

6.2.8 Procedure

The experiment was divided into four phases: welcome phase, training phase, execution phase and a debriefing phase. In the welcome phase, subjects were brought into a test room of a usability laboratory. Here, they were told that the aim of the experiment was to test the usability of two calculators. Next, the subjects sat down in front of the PC, the electrodes were placed and the application was started. The experimenter left the test room and went to the observation room. In the training phase, subjects first read a paper introduction, which explained how building project costs were calculated. After reading, the subjects practised with the calculators. The practise consisted of the following operations: calculating the labour cost of one worker, storing an intermediate outcome in the calculator's memory, calculating the total labour cost by using stored intermediate outcomes, trying to remember memory locations, and performing a copying task including an interruption of the task, rating the experienced effort and waiting 45 seconds before

continuing. The training also shows that the task was finished once they produced the correct answer and pressed the *Finish* button. The subjects were instructed to solve the equation as quickly as possible. As a mnemonic for the memory locations, the instructions advised the subjects to mentally order the workers according to their memory locations and memorise the first letter of the workers. Subjects who managed to complete the training phase within 45 minutes entered the next phase. Those who took longer were excluded and received NLG 15 (€ 6.81). Subjects finishing the complete experiment received NLG 45 (€ 20.42).

In the execution phase, the subjects calculated the cost of six building projects and rated the mental effort expenditure. This phase started with the PC randomly assigning the subjects to an entry in the scheme. After this, the first building project description was given. Subjects were given 75 seconds to read a description. Next, a table with the memory locations and intermediate outcomes was presented for 40 seconds. When this time had passed, the memory locations disappeared, the table rows were again randomly ordered and the calculator became visible. After solving the requested equation correctly, the subjects rated the effort and rested 45 second before continuing with the next task. In the debriefing phase, subjects were asked to fill out the ease-of-use and satisfaction questionnaire. After this the subjects received their financial reward.

To prevent subjects from endlessly trying to solve a task, a threshold time of 40 minutes was set after which the experimenter would help the subjects. The threshold time was based on a pilot study of three subjects and was set as the average task time (11.5 minutes) plus three times the standard deviation (9.3 minutes). The experimenter entered the test room in cases where subjects started a new attempt at solving the equation after crossing this threshold time. The subjects received the algebraic format of the equation and were again left to let them finish the task on their own.

6.3 Results

Four subjects did not finish the training phase within the time set. Four new subjects replaced these subjects. A total of 8 females and 16 male subjects, between the ages of 19 and 25 years completed the experiment. Five subjects received help because they exceeded the threshold time and one of them twice. In all these cases, the subjects were solving a difficult equation, in three cases with the small display calculator and in two cases with the large display calculator.

6.3.1 Data preparation

Questionnaire

For ease of analysis, the averages of the six ease-of-use and two satisfaction questions for each of the six objects rated ($2 \times$ calculator overall, $2 \times$ Editor versions, $2 \times$ processor &

memory applied in a specific calculator) were taken as perceived ease-of-use and satisfaction measures. After the debriefing, one subject indicated not to have paid attention to the direction of the scale in the ease-of-use and satisfaction questionnaire. Therefore, the answers of this subject were regarded as missing. Table 6.1 shows the results of reliability analyses performed on this questionnaire. Each of the ease-of-use measures (six questions per component or calculator) had an acceptable reliability of 0.8 or more. Two satisfaction measures (two questions per component or calculator), on the other hand, were below the 0.7 – 0.8 minimal level often recommended (Landauer, 1997).

Table 6.1: Cronbach alpha derived from reliability analyses performed on the multiple-items scales per component or calculator.

Measure	Coefficient Alpha	
	Ease of use	Satisfaction
Large calculator	0.89	0.53
Small calculator	0.92	0.88
Editor of the large calculator	0.80	0.79
Editor of the small calculator	0.93	0.77
Processor and memory of the large calculator	0.80	0.63
Processor and memory of the small calculator	0.89	0.92

Heart-rate variability

Before the HRV could be extracted from the ECG signal several transformations were needed. The peak detector in the Acqknowledge application extracted the so-called R-peaks of the ECG signal, to create a time series of the heartbeats. The time series were pre-processed for possible artefacts such as missing or prolonged heartbeats and extra heartbeats (L.J.M.Mulder, 1988). The extra heartbeats were removed; whereas missing heartbeats were corrected by inserting extra beats based on a linear interpolation. For one subject, the corrections made in three conditions took up more than 5% of the series; therefore, the data were regarded as missing. Only the first five minutes were taken, ignoring the first 30 seconds because they were not corrected for possible artefacts. In this way, possible duration effects that could mask effects in the first 5 minutes were prevented because the average task time of the conditions varied between 5.9 minutes and 18.9 minutes.

The corrected time series of heartbeats were of an unequal interval, which is inherent in heart-rate variability. Therefore, a fifth order Lagrange interpolation was applied to obtain equidistant time series (G.Mulder, 1980). The next step was the transformation of the equidistant time series to modulation index series. This is the expression of each

sample as a percentage of the mean inter-beat-interval. This removes the possible effect the heart-rate may have on the heart rate variability (Dellen, Aasman, Mulder, & Mulder, 1985). Finally, the frequency spectrum was analysed with a Fast Fourier Transformation with a moving Hanning window of 100 seconds and a window shift of 5 seconds. Other effects on the heart rate variability such as body temperature regulation (area between 0.02 to 0.06 Hz) and respiration-related fluctuations (area between 0.15 to 0.5 Hz) were removed by only taking the area between 0.07 and 0.14 Hz, related with short-term blood pressure regulation, as the HRV 0.1 Hz band measure. A decrease in this 0.1 Hz band is related to an increase in mental effort (G.Mulder, Mulder, Meijman, Veldman, & Roon, 2000).

6.3.2 Statistical analyses

The experiment had a 2×3 design. However to reduce the complexity, the main hypothesis was analysed separately from the alternative ineffectiveness hypothesis. For the main hypothesis, analyses were conducted on a 2×2 design —editor (small or large) \times equation difficulty (easy or difficult)—, and for the alternative hypothesis, a separate analysis was conducted that only looked at the effect the Editor versions (small or large) had on the copying equation task.

Mental effort

The first statistical analysis looked for an effect of the Editor versions and the equation difficulty on mental effort. A doubly MANOVA was performed on the logarithmically transformed HRV 0.1 Hz band measure and the RSME score. The logarithmically transformation was applied to decrease the effect of extremely high values. The within-subject variables were the Editor version (2) and the equation difficulty (2). A significant main effect for the equation difficulty on mental effort was found ($F(2,21) = 14.57$; $p. < 0.001$). The same main effect was also found in the RSME score ($F(1,22) = 30.52$; $p. < 0.001$). The subject rated the required effort lower for an easy equation than for solving a difficult equation (Figure 6.7). An interaction effect between the equation difficulty and the Editor version was also found ($F(2,21) = 4.15$; $p. = 0.030$). This effect can again be found in the HRV ($F(1,22) = 8.01$; $p. = 0.010$). Figure 6.8 shows the means in the four conditions. For the small display calculator, the changeover from the easy to difficult equations seems to be associated with an increase in the HRV, whereas for the large display calculator a decrease is apparent.

Overall performance

The next analysis looked for effects of the Editor version and the equation difficulty in the overall performance measures. A doubly MANOVA was conducted on task time and

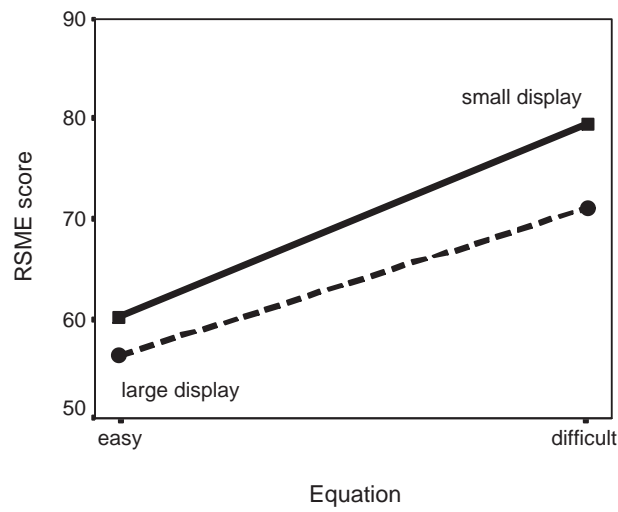


Figure 6.7: RSME score for the two editors given after performing an easy and a difficult equation.

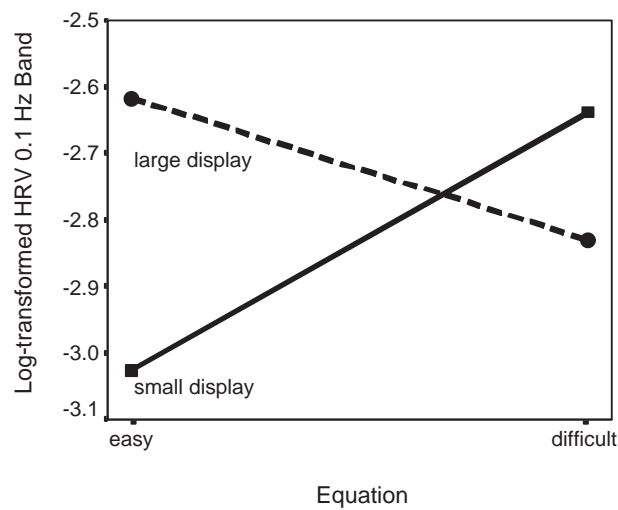


Figure 6.8: The logarithmically transformed HRV of the 0.1 Hz band for the two editors while performing an easy and a difficult equation. A decrease in HRV is interpreted as an increase in mental effort.

number of keystrokes. Both measures were first logarithmically transformed. The same within-subject variables were applied as before, Editor version (2) and equation difficulty (2). The analysis revealed a main effect for the equation difficulty ($F(2,22) = 33.00$; $p. < 0.001$), and again in both the task time ($F(1,23) = 42.08$; $p. < 0.001$) and the number of keystrokes ($F(1,23) = 16.58$; $p. < 0.001$). Inspection of the means showed that more time and keystrokes were needed in the case of a difficult equation. The doubly MANOVA revealed neither a significant main effect for the Editor version ($F(2,22) = 1.64$; $p. = 0.217$) nor a two-way interaction effect ($F(2,22) = 0.19$; $p. = 0.831$) between the equation difficult and the Editor version.

High-level control

To study the higher-level control strategy, the number of function requests, store requests and process requests sent to Processor & Memory interaction were taken as dependent measures for a doubly MANOVA. All measures were logarithmically transformed to reduce the influence of extreme values. The same within-subject design was used. The analysis revealed a significant main effect for the Editor versions. Subjects made significantly more process requests ($F(3,21) = 4.03$; $p. = 0.021$) when they had to calculate the building cost with the small display calculator than with the large display calculator (Figure 6.9).

Although no significant effect ($F(3,21) = 2.85$; $p. = 0.062$) for the equation difficulty was found in the multivariate analysis of the joint measure, significant effects were found in the univariate analyses of the number of process requests ($F(1,23) = 8.82$; $p. = 0.007$), function requests ($F(1,23) = 5.81$; $p. = 0.024$) and the store requests ($F(1,23) = 7.07$; $p. = 0.014$) separately. The subjects made more process and store requests (Figure 6.9 and 6.10) and requested more often the execution of a mathematical function when they solved a difficult equation than in the case of an easy equation. The analysis found no two-way interaction effect ($F(3,21) = 2.62$; $p. = 0.077$) in the joint measure. However, it found a two-way interaction effect ($F(1,23) = 6.81$; $p. = 0.016$) in the number of store requests. Inspection of the means (Figure 6.10) indicates that the subjects more often requested to store a result when they had to solve a difficult equation with a small display calculator than in all other conditions.

Effectiveness hypothesis

The alternative explanation for the variation in high-level control, was an effectiveness difference between the two Editor versions. A doubly MANOVA was conducted on the high-level performance measures and on the mental effort measures of the subjects performing a copying equation to examine this idea. The Editor version was taken as a within-subjects variable in this analysis. A significant effect ($F(5,18) = 3.00$; $p. = 0.038$) for the Editor versions was found by multivariate analysis on the joint measure. However, this effect was only found in univariate analyses of the HRV 0.1 Hz band ($F(1,22) = 9.02$; $p. = 0.007$) and the RSME ($F(1,22) = 8.65$; $p. = 0.008$) measures, but not in the number of

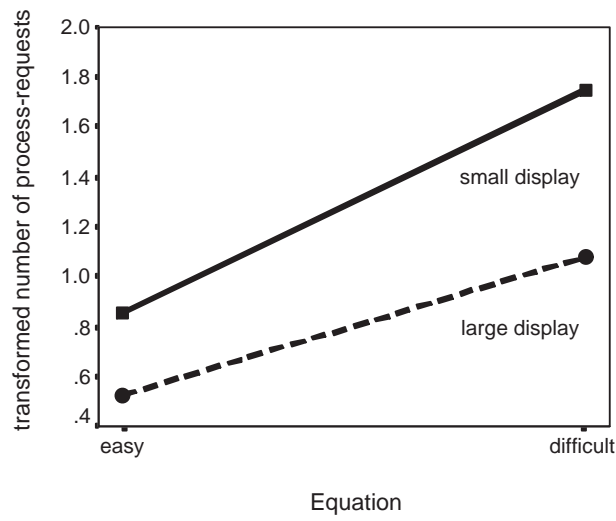


Figure 6.9: Log-transformed numbers of process-requests for the two editors while performing an easy and a difficult equation.

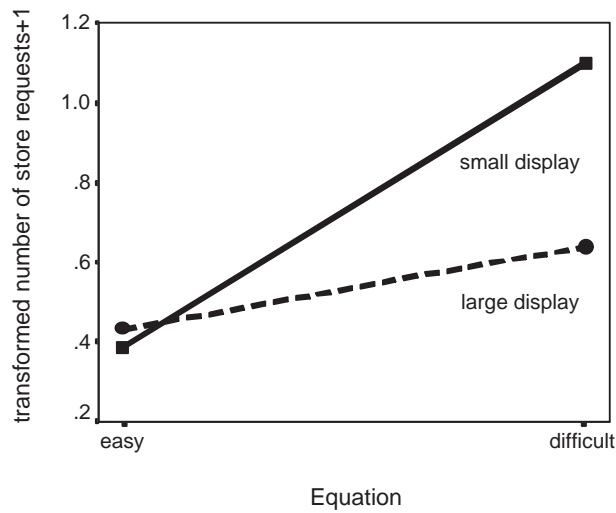


Figure 6.10: Log-transformed numbers of store-requests (plus 1) for the two editors while performing an easy and a difficult equation.

function requests ($F(1,22) = 2.06$; $p. = 0.165$), store requests ($F(1,22) = 1$; $p. = 0.328$), or process requests ($F(1,22) = 2.19$; $p. = 0.153$). Inspection of the means showed a higher RSME rating and lower HRV value when the subjects used the small display instead of the large display calculator. Furthermore, the standardised reception coefficient (SRC) for the two Editor versions was calculated for each of the different levels of equation difficulty (Table 6.2). All values were below 1. An SRC value greater than 1 would have suggested ineffectiveness.

Perceived usability

The averages of the subjective overall and component-specific usability measures were analysed for an effect for the Editor version only. A MANOVA was conducted on the perceived ease-of-use and satisfaction measures with the Editor version (2) as a within-subjects variable. The analysis revealed an effect for the Editor versions in the ease-of-use ($F(3,20) = 26.31$; $p. < 0.001$) and the satisfaction ($F(3,20) = 27.95$; $p. < 0.001$) measures. This effect was reflected in all measures separately (Table 6.3).

6.4 Discussion and conclusions

The results support the main hypothesis that mental effort creates a link between the Editor versions and the control strategy of the Processor & Memory interaction component. When solving a difficult equation with a small display calculator, subjects changed to a less efficient strategy and spent less effort in solving the equation. Although apparently reasonable to assume, the experiment did not demonstrate a causal relation between effort expenditure and the change in strategy. Mental effort was an intermediate variable, not under the direct control of the experiment, which is needed to make a causal inference. On the other hand, mental effort can never be directly manipulated, only the situation in which a person has to operate.

6.4.1 Interpretation results

Observed control

The benefit of the component-based usability testing framework becomes apparent when looking at the findings of the objective overall and component-specific performance measures. The overall performance measures did not reveal a main effect for the Editor version or an interaction effect between the Editor version and the equation difficulty. Apparently, the high-level strategy change was too subtle to be observed in the overall performance measures. The component-specific performance measures of Processor & Memory interaction component were more powerful. The strategy change was found in the number of

Table 6.2: Mean standardised reception coefficient values of the Editor interaction component for each level of equation difficulty.

Editor	Equation difficulty		
	Easy	Difficult	Copying
Small	0.78	0.55	0.76
Large	0.84	0.63	0.87

Note. All values are significantly smaller ($p < .01$) than 1.

Table 6.3: Results of multivariate and univariate analyses of variance with the answers on the ease-of-use and satisfaction questionnaire as dependent variables.

Measure	df 's	F	p	η^2
Ease-of-use				
Joint measure	3, 20	26.31	<0.001	0.79
Total calculator	1, 22	85.86	<0.001	0.80
Editor	1, 22	71.50	<0.001	0.77
Processor & Memory	1, 22	34.30	<0.001	0.61
Satisfaction				
Joint measure	3, 20	27.95	<0.001	0.81
Total calculator	1, 22	87.84	<0.001	0.80
Editor	1, 22	82.05	<0.001	0.79
Processor & Memory	1, 22	33.04	<0.001	0.60

times subjects sent a request to process an equation and to store the result. When the subjects were solving a difficult equation with the small display calculator, they more often stored intermediate outcomes than in the other conditions. Presumably, the subjects applied a so-called problem-reduction strategy (Halasz & Moran, 1983). This means breaking the problem into sub-problems, solving the sub-problem, storing them away and finally combining the intermediate outcomes into the overall result. In the other conditions, the subjects tend towards calculating the building cost within a single equation. The results of the process-requests suggest that they were less successful in doing this when working with the small display calculator than with the large display calculator.

Perceived control

Contrary to the objective overall measures, the analysis of the overall perceived ease-of-use and satisfaction did reveal an effect for the Editor version. The effect was also found in the separate rating of the Editor and in Processor & Memory interaction components. At first, it seems that Editor versions affected the subjects' attitude towards the Processor & Memory interaction component and the overall calculator. Although the relation between Editor version and the rating of the overall calculator is to be expected because the Editor interaction component is a part of the calculator, the relation between the Editor version and the rating of the Processor & Memory interaction component is less straightforward. Three explanations can be offered for this finding: (1) the subjects rated the Processor & Memory interaction component for their suitability in the context of the Editor; (2) the subjects made no clear distinction in their control experience of the two interaction components; (3) or the subjects gave socially desirable answers. The first explanation is in line with the main thesis of this chapter. The way interaction components affect one another may have a negative impact on their usability. The second explanation could mean that the subjects did not experience controlling two interaction components, or the given definition did not relate unambiguously to their object of control. In the previous studies (see chapter 4 and 5), subjects proved to be able to rate different interaction components, which means that they can make distinctions in their control experience. However, in chapter 4 it was clearly shown that other interaction components could have an impact on subjective component-specific measures. The third explanation should also be considered because this study was set up as a within-subject design. The subjects were confronted with the Processor & Memory interaction component in both calculators. Asking to rate them might have pushed the subjects into believing that they were expected to see a difference and they might have been led by their attitude towards the display size in doing so.

Mental effort

At first, the results obtained with RSME and HRV did not give a coherent picture. The perceived effort was rated higher for a difficult equation than for an easy equation, and

no effect was found for the Editor version. The results of the HRV did not show main effects, but only an interaction between Editor version and equation difficulty. Whereas the subjects spent less effort when solving a difficult rather than an easy equation with the small display, the opposite seems to be true when subjects solved them with the large display. Different results between RSME and HRV has been observed before (Zijlstra, 1993). The explanation was that the RSME could be conceived as an expression of subjective, or psychological costs, which people estimate according to two criteria: time and effort. HRV, on the other hand, only relates to mental effort. Since the task duration was indeed higher for difficult equations than for easy equations, it appears that the subjects took the task duration as the main estimation criterion in rating the psychological costs. Therefore, interpreting the RSME and HRV in this light, it appears that the subjects were able to solve the easy equation with the small display, although this required more mental effort than solving it with the large display calculator. When the subjects had to solve a difficult equation, subjects with large display spent more mental effort, whereas subjects with the small display changed to a less efficient strategy and ended up in spending less mental effort, although they perceived it as if they had spent more effort.

Refutation of alternative hypothesis

The results did not support the alternative ineffectiveness hypothesis. When the subjects had to solve the copying equation, no effects were found in the number of higher-level messages. If an effectiveness variation between the Editor versions was the root of the variations in the number of higher-level messages, it also had to be apparent in this very easy question. The absence of such an effect made the alternative hypothesis very unlikely. Indeed, the effort measures even showed that subjects spent more mental effort with the small display than with the large display calculator in this predominantly Editor-controlling task. Although all SRC values were smaller than 1, this does not suggest that all messages were intentionally sent upwards. However, these SRC values would certainly not suggest ineffectiveness.

6.4.2 Theoretical implications

Mental effort is a factor that makes the usability of the entire user interface involve more than only the usability of the separate components. This finding contradicts LPTs' claim that control loops are unaffected when lower-level layers are replaced as long as they provide the same messages services to the layer above it. The usability of the Processor & Memory interaction component depended on the version of the Editor interaction component. The small display editor may be very usable with a processor that automatically calculates intermediate outcomes and does not wait until an '=' button is pressed. However, preferring the small display editor to the large display editor is a mistake when it comes to the Processor & Memory interaction component used in this experiment.

The study shows that the relation between higher and lower control levels is more than only setting the reference value and receiving the higher-level feedback. Within PCT and LPT, two interpretations can be offered how mental effort connects the control levels with each other in this experiment. The first interpretation is that the lower-level control loop exhausted a limited mental resource, which was shared by the higher-level control loop. The second interpretation is that the lower-level control loop supported the higher-level in such a way that it required less mental effort to control.

Mental resources as intrinsic controlled variables

The concept of limited mental resource is used in limited mental capacity theories (Broadbent, 1958; Kahneman, 1973; Wickens, 1984), which state that the fundamental constraint that underlies all mental operations is a limited information-processing capacity. These theories suggest a mechanism that interferes with all control loops. The only mechanism within PCT that can do this is the reorganisation system (Powers, 1973). This system, responsible for altering the operations of all control loops, is an inherited control process that is concerned with keeping a set of critical variable near built-in reference conditions, such as nutrition and temperature. Although mental resources are not explicitly mentioned as one of these variables, it would explain the change in the higher-level strategy control as set in motion by the reorganisation system to bring the mental effort expenditure within intrinsic limits.

Besides memory capacity, the suggested interference effect could also be applicable to other types of resource sharing. The sharing could take place in the following dimensions: the stages of processing (perception, working memory, and cognition versus responding); the codes of perceptual, working memory and cognitive processing (verbal versus spatial); the input modalities (visual versus auditory); and response modalities (manual versus vocal) (Wickens, 1984). On the other hand, the inefficiency of some combination can of course be explained through masking. Overlaying windows makes it impossible for users to see the content of underlying windows and loud sounds can suppress other sounds.

Lower-level feedback to support higher-level control

Opposing this idea of limited capacity are theories (Allport, 1980; Logan, 1988; Neisser, 1976) that take skill as the limiting factor. Mental activities are conceived as a collection of acquired skills rather than the operation of a fixed mechanism. The choice for a control strategy depends on the subjects' skill of applying the strategy with all the interaction components. For a single-equation solution of the difficult equation with the small display, the subjects lacked the skill of keeping track of the intermediate realisation of the equation in their mind. The large display did not require the extra skill. The subjects could simply read this from the display.

The lower-level interaction component supported the higher-level interaction component, in the construction of a message it wanted to pass on, by sending upwards the feedback

message ‘equation formulated so far’ (Figure 6.1). For the large display, this meant the complete equation entered so far, whereas for the small display, this meant only the last value or operator. The Processor & Memory interaction component did not send this message; hence, the conclusion is that lower-level layers can offer higher-level layers support in formulating of their high-level actions. However, can this kind of message be qualified as E-feedback? E-feedback guides users in establishing expectations of what is achievable within a layer and guides users in selecting appropriate actions. The support can be interpreted as guiding the selection of the appropriate actions; however, these actions were of a higher-level layer. If this support is seen as another kind of feedback, is this only related to cases in which the absence of this support is accompanied by increase of mental effort or changes in higher-level strategies? Without an answer to this question, it seems only justified to propose a new kind of history message for now, which prevents high memory load. This message supports higher-level layers in constructing their lower-level messages by offering them information on the realisation of the message so far. The quality of this message affects the efficiency of higher-level layers.

6.4.3 Practical implications

When designing a user interface, attention should be given to situations where users have to cope with a heavy memory load. Especially in these situations, lower-level interaction components should support users when they establish complex higher-level messages. Information needed to accomplish a task should preferably be distributed more to the system than to the user side of the interaction (Zhang & Norman, 1994). In that case, users no longer have to cope with the memory demand of keeping this information active. In this experiment the memory load was pushed to the extreme by interrupting the task and the memorisation of the memory location places. More practical situations can also be imagined; for example, when users are easily interrupted as they dial a telephone number. In such a situation it helps to use a telephone where the entered number is displayed, rather than only giving sound feedback to indicate that a digit is entered.

Since the mental effort affect the users’ performance, it is better to perform a usability test in a realistic environment, which requests the same mental effort expenditure. An indication for a strategy shift is observed when a decrease in user performance coincides with a decrease in mental effort expenditure. Expanding a usability test with a mental effort measure is advisable, because component-specific measures may be affected by mental effort. In this study the physiological HRV measures was capable of measuring only the mental effort, whereas the RSME was more correlated with the task duration. To replace the physiological measure, another subjective measure is needed that is less correlated with the task duration and only measures mental effort. Another point of interest for usability tests, is the low reliability of the component-specific satisfaction measures in this experiment. Extending the two satisfaction questions may help to increase the reliability of this measure.

Finally, the experiment also shows the benefit of dividing the stream of messages received by an interaction component according to its content (store or process request) and analysing it separately. As was already noted in chapter 2, it may provide more insight into the layered user-system interaction.

Chapter 7

Discussion and conclusions

7.1 Recapitulation

As user interfaces are assembled out of a set of components, suitable theories are needed to create usable products this way. The Layered Protocol Theory (LPT), a user-system interaction theory, fits the component-based software engineering approach of user interfaces. LPT describes the user interface as consisting of several layers of interaction. The main tenet of LPT is that the purpose of any behaviour is to control perception. Therefore, the theory considers the users' desire to perceive a system state that matches their internal reference value as the main purpose for user-system interaction. In this theory, usability can be regarded as the ease with which users control a system. Central concepts in the theory are the perceptual-control loop and the accumulation of these control loops. The accumulation creates multiple layers on which interaction takes place. Each control loop is regarded as an exchange of messages, which has to obey a protocol.

Since the users try to control the system on several layers of interaction, the system usability might also be decomposed into the usability of the individual layers. In practice this would mean that the usability of interaction components, identified in the software architecture, are assessable. This possibility is very welcome since the creation and deployment of components is allocated to different software engineering processes. Usability evaluation of a component in the creation process would be more efficient than testing the usability of the component each time it is deployed in an application. Usability evaluation in the deployment process would not be necessary if the usability of an entire application only depends on the usability of the individual components. The latter is the case according to LPT because control loops are unaffected when lower-level layers are replaced as long as they provide the same message services to the layer above it. Until now, LPT has only been used to analytically evaluate the user interface of products. However, LPT is also suggested to provide a basis to evaluate the usability of components empirically.

This thesis had the following main research question: *Is usability compositional?* The question basically had two underlying questions:

1. Whether and how the usability of components can be tested empirically.
2. Whether and how the usability of components can be affected by other components.

In this final chapter the main conclusions drawn (Table 7.1) in the previous chapters are again presented with their limitations. The implications of the research findings both theoretical and practical are discussed. The chapter concludes by looking at possible further research topics.

Table 7.1: Main conclusions drawn in this thesis.

No	Conclusions
1	The Layered Protocol Theory provides a basis for empirical evaluation of the user interface for different layers of interaction.
2	An analysis of the number of messages received by an interaction component is as effective or more effective in determining usability variations between versions of both lower and higher-level interaction components than an analysis of the number of keystrokes in cases where components operate independently.
3	Inconsistency can cause interaction components to affect each other's usability.
4	Mental effort can cause interaction components to affect each other's usability.

7.2 Main conclusions and limitations

7.2.1 Component-based usability testing

The first main conclusion that can be drawn is that LPT indeed provides a basis for empirical evaluation of the user interface at different layers. A framework was established for component-based usability testing. The framework allows usability evaluation of elementary units within a device that receive and send signals to and from users, so users can perceive and alter the units' physical state. If a unit, called an interaction component, is part of a higher-level control loop, the interaction is mediated by lower-level interaction components. The testing framework supports two testing paradigms, the single version and the multiple versions testing paradigm. In the single version testing paradigm, only

one version of each interaction component is tested. The focus then is to identify interaction components that hamper the overall usability of the user interface. In the multiple versions testing paradigm, different versions of interaction components are compared with each other. The question in this case is which version has the highest usability.

The number of messages a component receives was proposed as the objective component-specific performance measure (chapter 2 and 3). This measure reflects the users' effort to control their perception of a component. Each message is an expression that users are unsatisfied with the current system state they perceive and that they spend effort to match it with their reference value. When assuming that the amount of effort to send a message is similar for each message, the messages can simply be added up to create a performance measure. If this is not the case, individual effort values can be assigned to each message as is done in the single version testing paradigm.

Special attention could be given to situations where the minimal numbers of messages required to perform that task differ for each version of the component. The analysis can be done on the number of messages made in addition to the minimal number by subtracting the minimal number from the observed one as was sometimes done in chapter 4 and 5. This corrected number presents the performance relative to the optimal performance. However, the absolute number would be more preferable when choosing the most usable version of a component. After all, what counts in a usability test is how much effort users have to spend on controlling a component, and not how this relates to the amount of effort an ideal user would need to perform the same task.

The study described in chapter 4 shows that an objective component-specific performance measure is as effective or more effective in determining usability variations between versions of both lower and higher-level interaction components than overall performance measures in cases where components operate independently. The power of this component-specific measure comes from the reduction in statistical variance by limiting its focus to one control loop, and, consequently, locking out the variance caused by the users' effort to control other components. In cases where components are more dependent, limiting the focus to one control loop might be counterproductive. In theory, objective overall performance measures might then be more powerful since they take into account all control loops that might be affected.

For the single version testing paradigm, the number of messages a component receives is set against the performance of an ideal user and is corrected for control effects of lower and higher-level components (chapter 3). An effort value is assigned to each message received. These effort values are based on the effort value of the lowest-level messages that are linked to higher-level messages. At the lowest-level layer, weight factors of an arbitrary unit are assigned to the messages, which present the user effort value to send a single lower-level message. The measure has been shown to correlate well with overall and component-specific usability measures and allows evaluators to order the components on their usability improvement potential (chapter 4). However, when it comes to applying this measure, it is presumed that components only receive messages that were sent with

the intent of controlling the components. The Standardised Reception Coefficient was introduced to help evaluators check whether messages were sent unintentionally as a side effect of lower-level control loops, which is labelled as an ineffectiveness problem.

The component-specific ease-of-use and satisfaction measures were obtained through a questionnaire. These component-specific measures were not found to be more effective in determining perceived ease-of-use or the satisfaction variations between versions of a component than their overall counterparts in the case of the multiple versions testing paradigm (chapter 4). The component-specific measures were expected to be more powerful because the specific questions could assist users in the recall of their control experience of individual interaction components (chapter 3). However, in the experimental set-up, subjects received both component-specific and overall questions in an randomly ordered sequence. The recollection triggered by the component-specific questions might have influenced the users when answering the overall questions. The benefit of component-specific ease-of-use and satisfaction measures lies in the single version testing paradigm. Here, evaluators can use them to compare the interaction components with each other, a quality overall measures do not possess in the single testing paradigm since they could only offer overall assessments of the single device.

The testing framework also has its limitations and evaluators should know them when interpreting results obtained by the testing framework (Gray & Salzman, 1998). First of all, ecological validity is always a point of concern when it comes to empirical evaluations because evaluations often affect the situation and working practices under consideration (Parsons, 1974). In the current framework participants receive a description of the goal they have to achieve as quickly as possible. Evaluators are responsible for setting goals that users normally would set themselves. Conclusions about the usability are only as valid as the goal given. Furthermore, as evaluators have to select achievable goals, the framework can not escape the “I know it can be done or you wouldn’t have asked me to do it” bias (Cordes, 2001). This limits the framework. The users’ ability to find out about the product’s capabilities and limitations can not be tested since they are already presented in the task instruction. Besides the bias for the goal setting, the speed constraint will not always be an important value of the interaction. However, it lowers the users’ ability to compensate for bad usability, and therefore, increases the power of the method. Although all the studies were conducted in a usability laboratory, the framework did not impose this. On the contrary, the framework lends itself for usability evaluations in the natural interaction environment. The objective performance measures are non-intrusive and the perceived ease-of-use and satisfaction measures are only obtained after the interaction.

The ultimate criterion for effectiveness of a usability evaluation method is how well the method helps evaluators discover *real* usability problems. Or phrased in a more general question, has it something to say about what people do in ‘real’, culturally and economically significant situations? This is a question often posed when it comes to cognitive theories (e.g. Hoc, 2000; Kaptelinin, 1996; Neisser, 1976). A usability problem is ‘real’ if it is a predictor of a problem that users will encounter in real work-context usage and that will have an impact on usability. In this study, the ‘reality’ was obtained by seeding *known*

usability problems into the prototypes. This approach has been criticised (Hartson, Andre, & Williges, 2001) as a standard approach to test analytic usability evaluation methods because it heavily depends on the researchers' skill to shape a problem in the prototype. Furthermore, the prototype is designed with the intent of testing and not of using, putting ecological validity in doubt. A more common approach is to take a product used in real life, with real-life usability problems, and to compare the results of analytic and empirical methods with each other.

Still, 'seeding' seems a more appropriate approach to evaluate the testing framework at this stage. Otherwise, it would require extensive research to find a device in real life with components that had real-life usability problems and also satisfied the demands of the experimental evaluation study described in chapter 4 (a troubling lower-level and higher-level component and a troubling component that effected the components in higher-level layers). Furthermore, the testing framework already is an empirical method, and other component-specific measures to compare with are not available at this stage. Next, the conclusions drawn agree with the conclusions derived from the commonly used empirical overall measures. Finally, the experiment was conducted on high fidelity prototypes, which were implemented on a computer, and closely emulated the E- and I-feedback of an actual product, including both visual and auditory system feedback.

Knowledge about the limitations of the range of usability problems the framework can handle is also valuable for evaluators. The experiments show that the framework can handle usability problems caused by the lack of understandable feedback, cognitive complexity, inconsistency and mental effort. The framework is based on the control of perception. Usability problems not related to the control of perception are not dealt with, such as health effects, both mental (e.g. stress) and physical (e.g. repetitive strain injury).

7.2.2 The independence of the component's usability

The other main conclusion that can be drawn is that the usability of interaction components can be affected by other interaction components. The usability of the entire product can not be predicted solely on the usability of the individual interaction components. The results of the experiments described in chapters 2 and 4 did not show that components affect each other's usability. However, inconsistency and mental effort are factors that make interaction components reduce users' ability to control another interaction component. The study described in chapter 5 showed that inconsistency could cause interaction components in the same layer or in higher layers to activate an inappropriate mental model for other interaction components. The likelihood that a mental model will be activated and applied to another interaction component, seems to depend on how powerful (often or recently applied) the mental model is. Inconsistency between the application domain and an interaction component was not found to affect the interaction component's usability. Whether this only was the case in this experiment or can be generalised, is a topic for further research. However, the study did show that the application domain had an effect

on the users' understanding of the functionality the component provides. The experiment described in chapter 6 showed that mental effort could link the control of higher-level interaction components to lower-level interaction components. A poor implementation of the lower-level layers can force users to adopt less efficient higher-level control strategies to cope with a mentally over-demanding task.

With respect to ecological validity, the situations created in the experiments of chapter 5 and 6 may be too extreme to occur in real life. The experiments primarily aimed at demonstrating that these factors do indeed exist. Taking a less extreme situation would give an unclear answer if no effect was found. However, tasks that are interrupted by other tasks, the applications used, and the goals given were not uncommon in real life, and some problematic component combinations even have direct representatives in applications used nowadays.

7.3 Implications

7.3.1 Theoretical implications

The conclusions drawn about the testing framework still strengthen LPT and the Perception Control Theory (PCT) as theories that can be successfully applied to the area of user-system interaction. The study illustrates the ability to determine from observation whether users successfully controlled their perception at the different layers of interaction. The theories also seem to set a framework for obtaining information how the users perceived their control in each of the different layers. Then again, the conclusions also indicate that control loops should no longer be seen as operating completely independently from each other, as posed in LPT and PCT. Memory emerges as an important connecting factor, for reasons of content as well as mental demand.

Strictly speaking, the relationship between control loops consists of setting reference values and of ultimately receiving the feedback messages. However, the study illustrates that memory load can force users to change higher-level control strategies when confronted with large mental demands. At first, mental effort has no explicit place in LPT or PCT. Still, the following two interpretations can be offered within the boundaries of these theories: memory support of lower-level layers for the construction of higher-level messages, or mental effort as an intrinsic reference signal that directs a reorganisation system. The difference between the two explanations comes down to the allocation of the mental effort in the control loop itself or in lower-level control loops. In the first interpretation, lower-level layers can support higher-level layers in the construction of a message it wants to pass on by sending feedback about the message's part sent so far. This lower-level mnemonic device reduces the mental effort put into the higher-level control loop. In the second interpretation, the relation between control loops is made by a reorganisation system (Powers, 1973). This system, responsible for altering the operations of all control loops, is an inherited control process that is concerned with keeping a set of critical variables near built-in

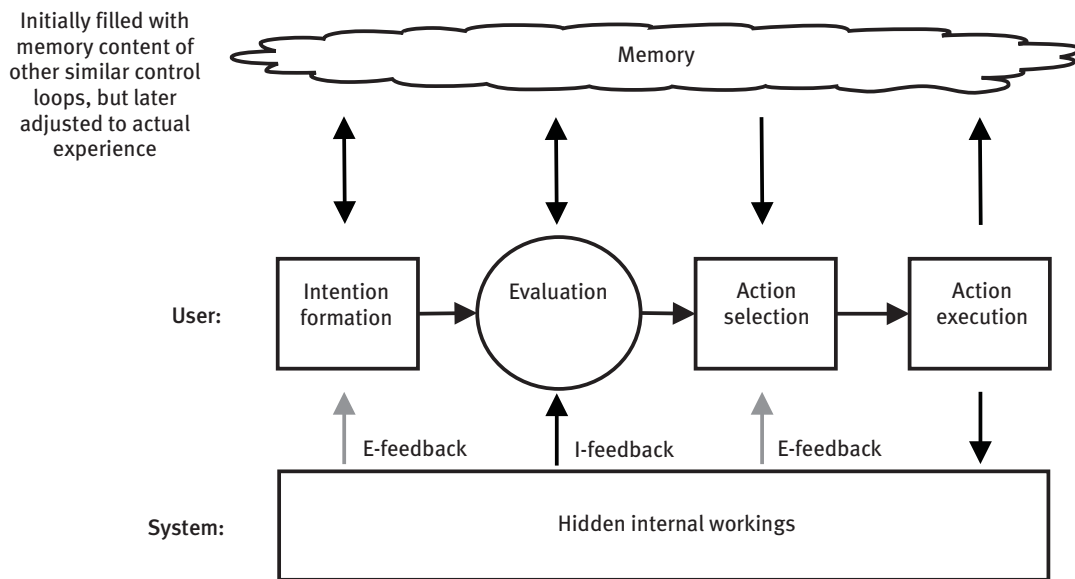


Figure 7.1: In the perceptual-control loop, memory helps with interpretation and creating expectations based on prior actions and feedback.

reference conditions, such as nutrition, temperature, and perhaps also mental resources. When there is a difference between sensed intrinsic state and the intrinsic reference signals, a reorganisation process is set in motion, which stops when the difference is close to zero. The strategy change observed in higher-level layers can be seen as the reorganisation process triggered when lower-level control loops exhaust mental resources.

The other argument against the independence claim is also related to memory. In its initial deployment, a control loop may depend on the memory recordings of other control loops. For the creation of control loops, Powers (1992) suggests a random reorganisation process, independent of other control loops. He suggested randomness (or noise) to explain how even the first new control loops come into being when there are no memory recordings to rely on. Still, this study illustrates that when new control loops show some similarity, users apply their experience from other control loops. ‘Similarity’ is defined here by whatever memory recordings are activated in this new situation. Memory recordings from other control loops may be copied or directly used as a base to start from in a new control loop. Memory in a control loop plays an essential role in the users’ ability to imagine the effect actions may have on perception (Powers, 1973). The current perception can be uncoupled from the control loop, while memory feeds the control loop old perceptions that are associated with actions. This ability is precisely what a mental model would offer. Without proposing what is recorded and its exact place in the perceptual-control loop, memory clearly guides users to alter their perception to their liking (Figure 7.1). However, applying memory in the wrong situation may get them into trouble. Especially when recordings are not made in the actual deployment of the control loop, but copied

from other seemingly similar control loops.

7.3.2 Practical implications

The research leads to a component-based usability testing framework, but also illustrates that components should not be designed, deployed and evaluated entirely independently of the other interaction components in user interfaces. Within the component-based software engineering approach, the creation and deployment processes are separate. Therefore, two kinds of usability evaluations are advised: one focusing on interaction components in isolation, the other focusing on the interaction component within the context of the particular application. As part of the creation process, the emphasis would be on the usability of the functionality offered by the interaction component. New interaction components could be evaluated in a test bed that presents potential future applications. The multiple versions testing paradigm would be most suitable in this environment. Different versions of a component could be compared with each other, and the most usable version could be shipped off to the component repository. Evaluation in this process is an efficiency step because it would affect many applications at once; usability problems related to the individual nature of the interaction component are already eliminated before components are deployed in applications.

As early as in the creation process, designers should be attentive to potential compromising factors that can turn against the usability later on in the deployment process. Two important factors are consistency and mental effort. To reduce future inconsistencies a set of interaction components should be envisaged that would be used together with the interaction component under construction. If this is not possible, designers may try to design according to some design rules; a style guide for example. In the deployment process, developers should check the anticipated environment or the applied design rules to see whether there are conflicting situations. In addition, the rules, but also the set of future components, can guide the design of a good test bed. Designers should also be cautious about creating interaction components that require a large amount of mental effort to operate, since the mental load may have a negative side effect on the operations of other interaction components as well. The search of interaction components that reduce the memory load of other interaction components is also effective. For example, interaction components could show what users have already done so far, so they do not have to rely on their memory when they need an overall view to proceed.

The usability of an entire product can not always be predicted from the usability of the individual interaction components. Therefore, a usability test, as part of the deployment process, should estimate the usability of the entire product and looks for usability problems in the context of the whole application. The single version testing paradigm fits well into this process, at least, if developers do not want to compare different versions of an interaction component in their application. At this point, evaluators should realise that the source of a usability problem can lie outside an interaction component or be a combination

with another interaction component. Measuring the mental effort may give more insight into this.

For user interfaces that are not developed according to the component-based engineering approach, the testing framework can still be useful once the control loops are identified. This would mean reverse engineering of a suitable compositional structure of the user interface. When this is established, the subjective component-specific perceived ease-of-use and the satisfaction measures can be obtained. Since there probably is no real message exchange between the components that were identified afterwards, objective component-specific performance measures can only be obtained by simulating the message exchange. Lower-level messages, such as keystrokes, should be fed into a conversion process that simulates the message exchange between the components, which is again recorded in a log file. The benefits of applying the testing framework are the statistical power of the objective component-specific measure, and the ability to study the individual component of the user interface directly instead of for its impact on the overall usability. The latter is especially useful within the single version testing paradigm where the individual components can not be assessed on the basis of overall usability measures.

7.4 Future work

The proof of the pudding is in the eating. The component-based usability testing framework seems promising. However, the real benefit will only become apparent when actual designers and developers put it into practice and the usability of the final product is assessed (Stanton & Young, 1999). It will then become visible how much extra effort and money is involved and how it fits in with normal working routines. Comparing usability predictions, made with the testing framework and field assessment of interaction components, may also help in establishing a better picture of the framework's effectiveness. This again, relates to a still open, more general question about the effectiveness of laboratory-based usability tests on the whole (Hartson et al., 2001). A field-based usability evaluation framework for components would do more justice to the ecological validity, putting the study in an social, organisational, and cultural context, in relation to goals, plan and values of users (Kaptelinin, 1996). Although the perceived ease-of-use and satisfaction measure could be adopted, the objective measure requires external goal-setting and time limitation. Taking the users' own goal setting into the evaluation creates a more comprehensive usability view. Furthermore, field-based studies on components open the door to long-term usability studies, which are rarely conducted (see for exceptions e. g. Nes & Itegem, 1990; Petersen, Madsen, & Kjær, 2002; Thomas, 1998). A component would be a more practical study object than an entire application since components have a longer life span.

In the experiment, subjects gave answers that were influenced by the usability of other interaction components. This led to recurring questions like, how does the observed control relate to the perceived control, and, what do users think they are controlling? Although according to LPT, the user-system interaction can be divided into several layers, it still

is undecided whether users make a clear distinction in their control experience of the different interaction components. If this were the case, LPT could also be a framework for the way in which users consciously perceive their interaction with a system. One step further would be an attempt to relate the users' utterances made in a usability test to the components in the user interface. This could form the basis for another component-specific usability measure, one that does not require extra coding in the prototype, and would fit the Thinking Aloud Protocol method often applied in usability tests. The action identification theory (Vallacher & Wegner, 1987) might give some ideas for a measure. The theory states that users will think in terms of higher levels of identification when the task is easy, but move to lower levels if their action proves difficult to maintain with higher-level identities in mind. An additional question, about users' perception of a component, can be put forward when an interaction component is in fact a composition of multiple smaller interaction components. Do users perceive a compound component as a coherent identity with the same functionality in different applications? If this is the case, developers can start building compound components that can be reused in different applications. If not, developers should construct user interfaces only by using basic interaction components.

Bibliography

- Allport, D. A. (1980). Attention and performance. In G. Claxton (Ed.), *Cognitive psychology: new directions* (p. 112-153). London: Routledge & Kegan Paul.
- Aykin, N. (1994). Software reuse: A case study on cost-benefit of adopting a common software development tool. In R. G. Bias & D. J. Mayhew (Eds.), *Cost-justifying usability* (p. 177-202). London: Academic Press.
- Bailey, B. P., Konstan, J. A., & Carlis, J. V. (2001). The effects of interruptions on task performance, annoyance, and anxiety in the user interface. In M. Hirose (Ed.), *INTERACT'01, IFIP TC.13 International conference on human-computer interaction* (p. 593-601). Amsterdam: IOS Press.
- Beaudet, D. B., & Williges, R. C. (1988). The role of screening studies in sequential research designs. In *Proceedings of the human factors society 32nd annual meeting* (p. 1174-1178). Santa Monica, CA: Human Factors Society.
- Blanchard, H. E., Lewis, S. H., Ross, D., & Cataldo, G. (1993). User performance and preference for alphabetic entry from 10-key pads: Where to put Q and Z? In *Proceedings of the human factors and ergonomics society 37th annual meeting* (p. 225-229). Santa Monica, CA: Human Factors and Ergonomics Society.
- Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *The unified modeling language user guide*. Amsterdam: Addison Wesley.
- Bourbon, W. T., & Powers, W. T. (1999). Models and their worlds. *International Journal of Human-Computer Studies*, 50, 445-461.
- Broadbent, D. E. (1958). *Perception and communication*. London: Pergamon.
- Brodsky, R. (1991). Your evolving phone number. *American Heritage of Invention and Technology*, 6, p. 64.
- Card, S. K., Moran, T. P., & Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, 23, 396-410.

- Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction*. London: Lawrence Erlbaum.
- Chandler, D. (2002). *Semiotics: the basics*. London: Routledge.
- Chéry, S., & Farrell, P. S. E. (1998). *A look at behaviourism and perceptual control theory in interface design* (DCIEM No. 98-R-12). North York, (Ontario), Canada: Defence and Civil Institute of Environmental Medicine.
- Chin, J. P., Diehl, V. A., & Norman, L. K. (1988). Development of an instrument measuring user satisfaction of the human-computer interface. In *Conference proceedings on human factors in computing systems* (p. 213-218). New York, NY: ACM Press.
- Cnossen, F. (2000). *Adaptive strategies and goal management in car drivers*. Doctoral dissertation, Rijksuniversiteit Groningen, The Netherlands.
- Coleman, W. D., Williges, R. C., & Wixon, D. R. (1985). Collecting detailed user evaluations of software interfaces. In R. W. Swezey, T. J. Post, & L. B. Strother (Eds.), *Proceedings of the human factors society - 29th annual meeting* (p. 240-244). Santa Monica, CA: Human Factors Society.
- Cordes, R. E. (2001). Task-selection bias: A case for user-defined tasks. *International Journal of Human-Computer Interaction*, 13, 411-419.
- Coutaz, J. (1987). PAC, an object oriented model for dialog design. In H.-J. Bullinger & B. Shackel (Eds.), *INTERACT'87: 2nd IFIP international conference on human-computer interaction* (p. 431-436). Amsterdam: North-Holland.
- Coutaz, J., Nigay, L., & Salber, D. (1996). Agent-based architecture modelling for interactive systems. In D. Benyon & P. Palanque (Eds.), *Critical issues in user interface systems engineering* (p. 191-209). Berlin: Springer Verlag.
- Curren, M. T., & Harich, K. R. (1994). Consumers' mood states: The mitigating influence of personal relevance on product evaluation. *Psychology & Marketing*, 11, 91-107.
- Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13, 319-340.
- Dellen, H. J. van, Aasman, J., Mulder, L. J. M., & Mulder, G. (1985). Time domain versus frequency domain measure of heart-rate variability. In J. F. Orlebeke, G. Mulder, & L. J. P. Doornen (Eds.), *Psychophysiology of cardiovascular control* (p. 353-374). New York, NY: Plenum Press.
- Detweiler, M. C., Schumacher, M. C., & Gattuso, N. (1990). Alphabetic input on a telephone keypad. In *Proceedings of the human factors society 34th annual meeting* (p. 212-216). Santa Monica, CA: Human Factors Society.

- Docampo Rama, M. (2001). *Technology generations handling complex user interfaces*. Doctoral dissertation, Technische Universiteit Eindhoven, The Netherlands.
- Doll, W. J., Hendrickson, A., & Deng, X. (1998). Using Davis's perceived usefulness and ease-of-use instruments for decision making: A confirmatory and multigroup invariance analysis. *Decision Sciences*, *29*, 839-869.
- Driskell, J. E., & Olmstead, B. (1989). Psychology and the military: Research applications and trends. *American Psychologist*, *44*, 43-54.
- Duce, D. A., Gomes, M. R., Hopgood, F. R. A., & Lee, J. R. (1990). *User interface management and design*. Berlin: Springer-Verlag.
- Edwards, J. L., & Sinclair, D. (2000). Design intelligence: A case of explicit models and layered protocols. In M. M. Taylor, F. Néel, & D. G. Bouwhuis (Eds.), *The structure of multimodal dialogue II* (p. 249-269). Amsterdam: John Benjamins.
- Eggen, J. H., Haakma, R., & Westerink, J. H. D. M. (1993). Layered protocols in user interfaces for consumer equipment. In S. Ashlund, K. Mullet, & A. Henderson (Eds.), *INTERACT '93 and CHI '93 conference companion on human factors in computing systems* (p. 165-166). New York, NY: ACM Press.
- Eggen, J. H., Haakma, R., & Westerink, J. H. D. M. (1996). Layered protocols: Hands-on experience. *International Journal of Human-Computer Studies*, *44*, 45-72.
- Eijk, R. van. (2000). *Simulatie van een mobiele telefoon; requirement, specificatie, ontwerp en realisatie*. Unpublished bachelor's thesis, Fontys Hogescholen, Eindhoven, The Netherlands.
- Engel, F. L., Goossens, P., & Haakma, R. (1994). Improved efficiency through I- and E-feedback: A trackball with contextual force feedback. *International Journal of Human-Computer Studies*, *41*, 949-974.
- Engel, F. L., & Haakma, R. (1993). Expectations and feedback in user-system communication. *International Journal of Man-Machine Studies*, *39*, 427-452.
- 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*, 489-520.
- Farrell, P. S. E., & Semprie, M. A. H. (1997). *Layered protocol analysis of a control display unit* (DCIEM No. 97-R-70). North York (Ontario), Canada: Defence and Civil Institute of Environmental Medicine.
- Foley, J. D., & Dam, A. van. (1983). *Fundamentals of interactive computer graphics*. Amsterdam: Addison-Wesley.

- Freudenthal, T. D. (1998). *Learning to use interactive devices; Age differences in the reasoning process*. Doctoral dissertation, Technische Universiteit Eindhoven, The Netherlands.
- Gediga, G., Hamborg, K.-C., & Düntsch, I. (1999). The IsoMetrics usability inventory: an operationalization of ISO 9241-10 supporting summative and formative evaluation of software systems. *Behaviour and Information Technology*, *18*, 151-164.
- Gillie, T., & Broadbent, D. E. (1989). What makes interruptions disruptive? A study of length, similarity, and complexity. *Psychological-Research*, *50*, 243-250.
- Gray, W. D., & Salzman, M. C. (1998). Damaged merchandise? A review of experiments that compare usability evaluation methods. *Human-Computer Interaction*, *13*, 203-261.
- Green, M. (1983). Report on dialogue specification tools. In G. E. Pfaff (Ed.), *User interface management systems* (p. 9-19). Berlin: Springer-Verlag.
- Grudin, J. (1989). The case against user interface consistency. *Communications of the ACM*, *32*, 1164-1173.
- 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*, 557-570.
- Hahn, J. (2001). The dynamics of mass online marketplaces: a case study of an online auction. In *Proceedings of the SIGCHI conference on human factors in computing systems* (p. 317-324). New York, NY: ACM Press.
- Halasz, F. G., & Moran, T. P. (1983). Mental models and problem solving in using a calculator. In A. Janda (Ed.), *Proceedings of 1983 conference on human factors in computing systems* (p. 212-216). New York, NY: ACM Press.
- Hartson, H. R., Andre, T. S., & Williges, R. C. (2001). Criteria for evaluating usability evaluation methods. *International Journal of Human-Computer Interaction*, *13*, 373-410.
- Heineman, G. T., & Councill, W. T. (2001). *Component-based software engineering: Putting the pieces together*. London: Addison-Wesley.
- Herrnstein, R. J. (1967). Introduction to J. B. Watson. In *Behavior; An introduction to comparative psychology* (p. xi-xxx). New York, NY: Holt, Rinehart and Winston.
- 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* (p. 88-94). Sydney: Ergonomics Society of Australia.

- Hertzum, M., & Jacobsen, N. E. (2001). The evaluator effect: A chilling fact about usability evaluation methods. *International Journal of Human-Computer Interaction*, *13*, 421-443.
- Hilbert, D. M., & Redmiles, D. F. (2000). Extracting usability information from user interface events. *ACM Computing Surveys (CSUR)*, *32*, 384-421.
- Hilbert, D. M., & Redmiles, D. F. (2001). Large-scale collection of usage data to inform design. In M. Hirose (Ed.), *INTERACT'01, IFIP TC.13 International conference on human-computer interaction* (p. 767-768). Amsterdam: IOS Press.
- Hoc, J.-M. (2000). Toward ecological validity of research on cognition. In *Ergonomics for the new millennium: Proceedings of IEA 2000/HFES 2000 congress* (Vol. 1, p. 549-552). Santa Monica, CA: Human factors and Ergonomics Society.
- Hockey, G. R. J. (1997). Compensatory control in the regulation of human performance under stress and high workload: A cognitive-energetical framework. *Biological Psychology*, *45*, 73-93.
- Holleman, G. (1999). User satisfaction measurement methodologies: Extending the user satisfaction questionnaire. In H.-J. Bullinger & J. Ziegler (Eds.), *Human-computer interaction: Ergonomics and user interfaces, proceedings of the 8th international conference on human-computer interaction* (Vol. 1, p. 1008-1012). Mahwah, NJ: Lawrence Erlbaum.
- Holyer, A. (1992). Top-down object-based user interface definition and design paradigms. In J. Gornostaev (Ed.), *Proceedings of East-West international conference on human-computer interaction* (p. 421-428). Moscow: ICSTI.
- ISO. (1994). *Information technology - open systems interconnection- basic reference model: The basic model* (ISO/IEC No. 7498-1). Geneva: International Organization for Standardization.
- ISO. (1998). *Ergonomic requirements for office work with visual display terminals (VDTs) Part 11. Guidance on usability* (ISO No. 9241-11). Geneva: International Organization for Standardization.
- Jacobson, I., Griss, M., & Jonsson, P. (1997). *Software reuse: Architecture, process and organization for business success*. New York, NY: ACM Press.
- John, B. E., & Kieras, D. E. (1996). The GOMS family of user interface analysis techniques: comparison and contrast. *ACM Transactions on Computer-Human Interaction*, *3*, 320-351.
- John, B. E., & Marks, S. J. (1997). Tracking the effectiveness of usability evaluation methods. *Behaviour and Information Technology*, *16*, 188-202.

- Jorna, P. G. A. M. (1985). Heart-rate parameters and coping process underwater. In J. F. Orlebeke, G. Mulder, & L. J. P. Doornen (Eds.), *Psychophysiology of cardiovascular control* (p. 827-839). New York, NY: Plenum Press.
- Kahneman, D. (1973). *Attention and effort*. Englewood Cliffs, NJ: Prentice-Hall.
- Kaptelinin, V. (1996). Activity theory: Implications for human-computer interaction. In B. A. Nardi (Ed.), *Context and consciousness* (p. 103-116). London: MIT Press.
- Kellogg, W. A. (1989). The dimensions of consistency. In J. Nielsen (Ed.), *Coordinating user interfaces for consistency* (p. 9-20). London: Academic Press.
- Kieras, D., & Polson, P. G. (1985). An approach to the formal analysis of user complexity. *International Journal Man-Machine Studies*, 22, 365-394.
- Kieras, D. E., & Bovair, S. (1984). The role of a mental model in learning to operate a device. *Cognitive Science*, 8, 255-273.
- Kirakowski, J., & Corbett, M. (1993). SUMI: The software usability measurement inventory. *British journal of educational technology*, 24, 210-212.
- Kobryn, C. (2000). Modeling components and frameworks with UML. *Communications of the ACM*, 43(10), 31-38.
- Kramer, J. J. (1970). Human factors problem in the use of pushbutton telephones for data entry. In *Human factors in telephony: 4th international symposium* (p. 241-258). Berlin: VDE-Verlag.
- 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, 27-49.
- Kreifeldt, J. G., & McCarthy, M. E. (1981). Interruption as test of the user-computer interface. In *Proceedings of the 17th annual conference on manual control, JPL publication 81-95* (p. 655-667). University of California.
- Landauer, T. K. (1997). Behavioral research methods in human-computer interaction. In M. G. Helander, T. K. Landauer, & P. V. Prabhu (Eds.), *Handbook of Human-Computer Interaction* (p. 203-227). Amsterdam: Elsevier.
- Lecerof, A., & Paternò, F. (1998). Automatic support for usability evaluation. *IEEE Transactions on Software Engineering*, 24, 863-888.
- Lewis, J. R. (1989). Pairs of latin squares to counterbalance sequential effects and pairing of conditions and stimuli. In *Proceedings of the human factors society 33rd annual meeting* (p. 1223-1227). Santa Monica, CA: Human Factors Society.

- Lewis, J. R. (1995). IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, 57-78.
- Lin, H. X., Choong, Y.-Y., & Salvendy, G. (1997). A proposed index of usability: A method for comparing the relative usability of different software systems. *Behaviour and Information Technology*, 16, 267-278.
- Logan, G. D. (1988). Automaticity, resources, and memory: Theoretical controversies and practical implications. *Human Factors*, 30, 583-598.
- Lohse, G. L., & Spiller, P. (1998). Quantifying the effect of user interface design features on cyberstore traffic and sales. In C.-M. Karat, A. Lund, J. Coutaz, & J. Karat (Eds.), *Conference proceedings on human factors in computing systems* (p. 211-218). Los Angeles, CA: ACM/Addison-Wesley.
- McIlory, M. D. (1979). Mass produced software components. In P. Naur, B. Randell, J. N. Buxton, & NATO science committee (Eds.), *Software engineering: concepts and techniques: proceedings of the NATO conferences, 1968 and 1969, Garmisch, Germany, and Rome* (p. 88-98). New York, NY: Mason/Charter.
- Meister, D. (1976). *Behavioral foundations of system development*. New York, NY: John Wiley & Sons.
- Meister, D. (1999). *The history of human factors and ergonomics*. London: Lawrence Erlbaum.
- Mulder, G. (1980). *The heart of mental effort: Studies in the cardiovascular psychophysiology of mental work*. Doctoral dissertation, Rijksuniversiteit Groningen, The Netherlands.
- Mulder, G., Mulder, L. J. M., Meijman, T. F., Veldman, J. B. P., & Roon, A. M. van. (2000). A psychophysiological approach to working conditions. In R. W. Backs & W. Boucsein (Eds.), *Engineering psychophysiology; Issues and applications* (p. 139-159). London: Lawrence Erlbaum.
- Mulder, L. J. M. (1988). *Assessment of cardiovascular reactivity by means of spectral analysis*. Doctoral dissertation, Rijksuniversiteit Groningen, The Netherlands.
- Myers, B. A. (1998). A brief history of human-computer interaction technology. *Interactions*, 5(2), 44-54.
- Nardi, B. A. (1996). Studying context: A comparison of activity theory, situated action models, and distributed cognition. In B. A. Nardi (Ed.), *Context and consciousness* (p. 69-102). London: MIT Press.

- NATO. (2001). *Visualisation of massive military datasets: human factors, applications, and technologies* (RTO-TR No. 030). Hull (Québec), Canada: North Atlantic Treaty Organization.
- Neerincx, M. A., & Greef, H. P. de. (1998). Cognitive support: Extending human knowledge and processing capacities. *Human-Computer Interaction, 13*, 73-106.
- Neisser, U. (1976). *Cognition and reality*. San Francisco, CA: W.H. Freeman and company.
- Nes, F. L. van, & Itegem, J. P. M. van. (1990). Hidden functionality: how an advanced car radio is really used. In F. L. Engel, D. J. Hermes, & J. B. O. S. Martens (Eds.), *IPO annual progress report 25* (p. 101-112). Eindhoven, The Netherlands: The Reproduction and Photography section of the Technische Universiteit Eindhoven.
- Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. In J. C. Chew & J. Whiteside (Eds.), *Conference proceedings on empowering people : Human factors in computing system: special issue of the SIGCHI Bulletin* (p. 249-256). New York, NY: ACM Press.
- Nigay, L., & Coutaz, J. (1991). Building user interfaces: organizing software agents. In Commission of EC, directorate-general: Telecommunication, Information, Industries and Innovation (Ed.), *Proceedings of annual Esprit conference* (p. 707-719). Office for Official Publications of the European Communities.
- Nigay, L., & Coutaz, J. (1993). A design space for multimodal systems: concurrent processing and data fusion. In S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, & T. White (Eds.), *Conference proceedings on human factors in computing systems* (p. 172-178). Amsterdam: ACM Press.
- Norman, D. A. (1981). Categorization of action slips. *Psychological Review, 88*, 1-15.
- Norman, D. A. (1984). Stages and levels in human-machine interaction. *International Journal of Man-Machine Studies, 21*, 365-375.
- Olsen, D. R., & Halversen, B. W. (1988). Interface usage measurements in a user interface management system. In *Proceedings of the ACM SIGGRAPH symposium on user interface software* (p. 102-108). New York, NY: ACM Press.
- Palay, A. J., Hansen, W. J., Kazar, M. L., Sherman, M., Wadlow, M. G., Neuendorffer, T. P., Stern, Z., Bader, M., & Peters, T. (1988). The Andrew toolkit - An overview. In *Proceedings of the winter USENIX conference* (p. 9-21). Berkeley, CA: USENIX Association.
- Parsons, H. M. (1974). What happened at Hawthorne? *Science, 183*, 922-932.
- Paternò, F. (2000). *Model-based design and evaluation of interactive applications*. London: Springer.

- Payne, S. J., & Green, T. R. G. (1989). The structure of command languages: An experiment on task-action grammar. *International Journal Man-Machine Studies*, 30, 213-234.
- Petersen, M. G., Madsen, K. H., & Kjær, A. (2002). The usability of everyday technology: emerging and fading opportunities. *ACM Transactions on Computer-Human Interaction*, 9, 74-105.
- Polson, P. G. (1988). The consequences of consistent and inconsistent user interfaces. In R. Guindon (Ed.), *Cognitive science and its applications for human-computer interaction* (p. 59-108). Hillsdale, NJ: Lawrence Erlbaum.
- 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, 741-773.
- Powers, W. T. (1973). *Behavior: the control of perception*. New York, NY: Aldine de Gruyter.
- Powers, W. T. (1992). *Living control systems II*. Gravel Switch, KY: The Control Systems Group.
- Powers, W. T. (1998). *Making sense of behavior*. New Canaan, CT: Benchmark Publications.
- Rauterberg, M. (1995). Four different measures to quantify three usability attributes: 'feedback', 'interactive directness' and 'flexibility'. In P. Palanque & R. Bastide (Eds.), *Design, specification and verification of interactive systems : proceedings of the Eurographics workshop* (p. 209-223). Wien: Springer.
- Reisner, P. (1993). APT: A description of user interface inconsistency. *International Journal of Man-Machine Studies*, 39, 215-236.
- Sanderson, P. M., & Fisher, C. (1994). Exploratory sequential data analysis: Foundations. *Human-Computer Interaction*, 9, 251-317.
- Sanderson, P. M., & Fisher, C. (1997). Exploratory sequential data analysis: qualitative and quantitative handling of continuous observational data. In G. Salvendy (Ed.), *Handbook of human factors and ergonomics* (2 ed., p. 1471-1513). Chichester: Wiley-Interscience.
- Shneiderman, B. (2000). Universal usability. *Communications of the ACM*, 43(5), 84-91.
- Sinclair, R. C., Mark, M. M., Moore, S. E., Lavis, C. A., & Soldat, A. S. (2000). An electoral butterfly effect. *Nature*, 408, 665 - 666.

- Smilowitz, E. D. (1995). *Methaphors in user interface design: An empirical investigation*. Unpublished doctoral dissertation, New Mexico State University, Las Cruces, NM.
- Snowberry, K., Parkinson, S. R., & Sisson, N. (1983). Computer display menu. *Ergonomics*, *26*, 699-712.
- Sperandio, J. C. (1971). Variation of operator's strategies and regulating effects on workload. *Ergonomics*, *14*, 571-577.
- Stanton, N. A., & Young, M. S. (1999). What price ergonomics? *Nature*, *399*, 197-198.
- Taylor, M. M. (1988a). Layered protocol for computer-human dialogue. I: Principles. *International Journal Man-Machine Studies*, *28*, 175-218.
- Taylor, M. M. (1988b). Layered protocols for computer-human dialogue. II: Some practical issues. *International Journal Man-Machine Studies*, *28*, 219-257.
- Taylor, M. M. (1989). Response timing in layered protocol: A cybernetic view of natural dialogue. In M. M. Taylor, F. Néel, & D. G. Bouwhuis (Eds.), *The structure of multimodel dialogue* (p. 159-172). Amsterdam: Elsevier science.
- Taylor, M. M. (1993). *Principals for intelligent human-computer interaction; A tutorial on layered protocol theory* (DCIEM No. 93-92). North York (Ontario), Canada: Defence and Civil Institute of Environmental Medicine.
- 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 Human-Computer Studies*, *50*, 521-555.
- Taylor, M. M., McCann, C. A., & Tuori, M. I. (1984). *The interactive spatial information system* (DCIEM No. 84-R-22). Downsview, (Ontario) Canada: Defence and civil institute of environmental medicine.
- Taylor, M. M., & Waugh, D. A. (2000). Multiplexing, diviplexing, and the control of multimodal dialogue. In M. M. Taylor, F. Néel, & D. G. Bouwhuis (Eds.), *The structure of multimodal dialogue II* (p. 439-456). Amsterdam: John Benjamins.
- The UIMS Tool developers workshop. (1992). A metamodel for the runtime architecture of an interactive system. *ACM SIGCHI Bulletin*, *24*(1), 32-37.
- Thomas, R. C. (1998). *Long term human-computer interaction: An exploratory perspective*. London: Springer.
- Vallacher, R. R., & Wegner, D. M. (1987). What do people think they're doing? Action identification and human behavior. *Psychological Review*, *94*, 3-15.
- Vollmeyer, R., Burns, B. D., & Holyoak, K. J. (1996). The impact of goal specificity on strategy use and the acquisition of problem structure. *Cognitive Science*, *20*, 75-100.

- Wickens, C. D. (1984). Processing resources in attention. In R. Parasuraman & D. R. Davies (Eds.), *Varieties of attention* (p. 63-102). London: Academic Press.
- Zhang, J., & Norman, D. A. (1994). Representation in distributed cognitive tasks. *Cognitive science*, 18, 87-122.
- Zijlstra, F. R. H. (1993). *Efficiency in work behaviour; A design approach for modern tools*. Doctoral dissertation, Delft University of Technology, The Netherlands.

Appendix A

Formal specification objective performance measure in the single version testing paradigm

A more informal description of this formal specification can be found in chapter 3 starting on page 49.

record received below : (sender, type, content, effort, time) (A.1)

record sent upwards : (recipient, type, content, effort, time) (A.2)

Data structure of a message

P : array[0.. S] of record received below (A.3)

Q : array[0.. T] of record sent upwards (A.4)

Data structure of an optimal (ideal-user) log file per interaction component

X : array[0.. M] of record receive below (A.5)

Y : array[0.. N] of record sent upwards (A.6)

Data structure of an observed (real-user) log file per interaction component

$$\text{effect}(x) \rightarrow \text{effect of message } x \text{ on higher level interaction component} \quad (\text{A.7})$$

$$\text{optimalEffort}(y) \rightarrow q.\text{effort} : q \in Q \wedge y.\text{type} = q.\text{type} \wedge \text{effect}(y) = \text{effect}(q) \quad (\text{A.8})$$

$$\text{receivedEffort}(K, L, n) \rightarrow (\text{Sum } k : k \in K \wedge (n = 0 \rightarrow k.\text{time} < L[n].\text{time}) \wedge (n > 0 \rightarrow L[n-1].\text{time} < k.\text{time} < L[n].\text{time}) : k.\text{effort}) \quad (\text{A.9})$$

Definition functions

$$(\forall i, j : 0 \leq i < M \wedge 0 \leq j < S : P[j] \text{ is lowest level message} \rightarrow P[j].\text{effort} = w[g] \wedge X[i] \text{ is lowest level message} \rightarrow X[i].\text{effort} = \text{optimalEffort}(X[i])) \quad (\text{A.10})$$

Basic effort-values have to be assigned to the messages at the lowest-level layer

$$(\forall i : 0 \leq i < T : Q[i].\text{effort} = \text{receivedEffort}(P, Q, i)) \quad (\text{A.11})$$

Assignment of the effort value to messages sent upwards in optimal task performance

$$\begin{aligned} & (\forall i : 0 \leq i < N : Y[i].\text{effort} = \\ & \text{if } (\exists j : 0 \leq j < T : Q[j].\text{type} = Y[i].\text{type} \wedge \text{effect}(Q[j].\text{type}) = \text{effect}(Y[i])) \\ & \quad \text{then Min}(\text{optimalEffort}(Y[i]), \text{receivedEffort}(X, Y, i)) \\ & \quad \text{else if } (\exists j : 0 \leq j < T : Q[j].\text{type} = Y[i].\text{type}) \\ & \quad \quad \text{then Min}((\text{Max } j : 0 \leq j < T \wedge Q[j].\text{type} = Y[i].\text{type} : Q[j].\text{effort}), \\ & \quad \quad \quad \text{receivedEffort}(X, Y, i)) \\ & \quad \quad \text{else receivedEffort}(X, Y, i)) \end{aligned} \quad (\text{A.12})$$

Assignment of effort value to message sent upwards in observed tasks

$$\text{Total Effort}_{\text{optimal}} = (\text{Sum } i : 0 \leq i < S : P[i].\text{effort}) \quad (\text{A.13})$$

$$\text{Total Effort}_{\text{observed}} = (\text{Sum } i : 0 \leq i < M : X[i].\text{effort}) \quad (\text{A.14})$$

Effort made by user to control interaction component

$$\text{User Effort}_{\text{observed}} = \frac{\text{Total Effort}_{\text{observed}}}{N} \times T \quad (\text{A.15})$$

Effort corrected for inefficiency higher-level layers

$$\text{Extra user effort} = \text{UserEffort}_{\text{observed}} - \text{Total Effort}_{\text{optimal}} \quad (\text{A.16})$$

Extra effort a user has to make on top of the effort an ideal user would make

```

i := 0
;do i ≠ N
  → | [if (∃j: 0 ≤ j < T: Q[j].type = Y[i].type) ∧
      (∀j: 0 ≤ j < T: Q[j].type = Y[i].type → effect(Q[j]) ≠ effect(Y[i])) → remove(Y,i)
      fi
      ;i:=i+1
  ]
od

```

Figure A.1: All messages observed that have a different effect on a higher-level interaction component than in optimal task execution should be ignored in the log file of the real users.

Appendix B

Standard questions used in the questionnaires

The six perceived ease-of-use questions taken from the Perceived Usefulness and Ease-of-Use (PUEU) questionnaire (Davis, 1989).

Learning to operate [name] would be easy for me.

Unlikely Likely
extremely quite slightly neither slightly quite extremely

I would find it easy to get [name] to do what I want it to do.

Unlikely Likely
extremely quite slightly neither slightly quite extremely

My interaction with [name] would be clear and understandable.

Unlikely Likely
extremely quite slightly neither slightly quite extremely

I would find [name] to be flexible to interact with.

Unlikely Likely
extremely quite slightly neither slightly quite extremely

It would be easy for me to become skillful at using [name].

Unlikely Likely
extremely quite slightly neither slightly quite extremely

I would find [name] easy to use.

Unlikely Likely
extremely quite slightly neither slightly quite extremely

The two satisfaction questions taken from the Post-Study System Usability Questionnaire (PSSUQ) (Lewis, 1995).

The interface of [name] was pleasant.

Strongly disagree Strongly agree
1 2 3 4 5 6 7

I like using the interface of [name].

Strongly disagree Strongly agree
1 2 3 4 5 6 7

Summary

Everyday, people are confronted with devices they have to control for all kinds of reasons. Designers hope that their creations can be controlled easily. For that purpose, they can turn to user-system interaction theories to guide them in design and evaluation. However, so far, no empirical methods have been developed to evaluate the usability that corresponds well with the increasingly popular approach of component-based software engineering. Instead of building a device from scratch, the component-based software engineering approach focuses on building artefacts from already made components (e. g. pop-up menus, radio buttons, and list boxes). The usability of components has not yet been assessed individually, but only for their impact on the overall usability (e. g. number of keystrokes, task duration, or questionnaires about the overall ease of use and satisfaction).

The Layered Protocol Theory (LPT) regards interaction as an exchange of messages between components and the user. LPT decomposes the user-system interaction into different layers that can be designed and analysed separately. It claims the possibility of component-specific usability evaluation. This is indeed very welcome since the creation and deployment of components is allocated to different processes in the component-based software engineering approach. Usability evaluation of a component in its creation process would be more efficient than testing the usability of the component each time it is deployed in an application. Usability evaluation in the deployment process is not even necessary if the usability of an entire application only depends on the usability of the individual components. The latter is the case according to LPT, because layers are unaffected when lower-level layers are replaced as long as they provide the same message services to the layer above it.

Until now, LPT has only been used to analytically evaluate the user interface of products. However, LPT is also suggested to provide a basis to evaluate the usability of separate components empirically. To do so the claim about the independence of the layers is essential, which however, has not yet been examined empirically. Therefore, the thesis has the following main research question: *Is usability compositional?* The question basically has two underlying questions:

1. Whether and how the usability of components can be tested empirically.
2. Whether and how the usability of components can be affected by other components.

The research was conducted in a series of laboratory experiments in which subjects had to perform tasks with prototypes of various user interfaces. The first experiment was conducted to search for an objective component-specific performance measure. In this explorative experiment, 80 university students operated a fictitious user interface. In a training session, the subjects received one out of eight instruction sets, which were created by providing or withholding information about three components. Before and after the subjects performed the tasks their knowledge about the components was tested. Furthermore, the message exchange between the components was recorded in a log file during the task execution. The results showed that the subjects' knowledge about a component affected the number of messages it received. This suggests that the number of messages a user interface component received can be taken as an objective component-specific performance measure. The measure indicates the users' effort to control their perception of the component. Each message is an expression that users are unsatisfied with the state of the system they perceive, and that they spend effort changing it to a perception they desire.

A framework has been established to test the usability of components, which is based on the findings from this explorative experiment. The testing framework supports two testing paradigms, a single version and a multiple versions testing paradigm. In the single version testing paradigm, only one version of each component is tested. The focus is on identifying components that hamper the overall usability of the user interface. In the multiple versions testing paradigm, different versions of components are compared with each other. The question in this case is which version has the highest usability.

For the single version testing paradigm, the number of messages a component receives is compared with the performance of an ideal user and is corrected for control effects of lower and higher-level components. An effort value is assigned to each message received. These effort values are based on the effort value of the lowest-level messages that are linked to higher-level messages. At the lowest-level layer, weight factors are assigned to the messages, which represent the user effort value of sending a single lower-level message.

The subjective component-specific measure for the ease of use and satisfaction were obtained through a standard usability questionnaire. These measures were expected to be more powerful than subjective overall measures, because the specific questions could assist users in the recall of their control experience of individual components.

In a second experiment, the framework was evaluated by comparing overall and component-specific usability measures for their ability to identify usability problems that were created in advance. Eight different prototypes of a mobile telephone were constructed by designing two versions of three components. The versions of each component were designed to differ clearly with respect to their usability. Eight groups of ten university students had to complete a set of tasks with one specific mobile telephone prototype. The results did not reveal subjective component-specific measures to be more effective in determining variations in the perceived ease of use or the satisfaction between versions of a component than their overall counterparts in the case of the multiple versions testing paradigm. This, however, could be an artefact of the experiment because both component-specific and

overall questions were given in a random order. The memory recall of the component-specific questions could have affected the overall questions as well.

The results of the second experiment did, however, show that an objective component-specific performance measure is more effective in determining usability variations between versions of both lower and higher-level components than overall performance measures in cases where components operate independently. The power of this component-specific measure comes from the reduction in statistical variance by limiting the focus to one component, and, consequently, locking out the variance caused by the users' effort to control other components. For the single version testing paradigm, the results showed the objective component-specific performance measure to correlate well with overall and subjective component-specific usability measures and to allow evaluators to order the components according to their potential to improve the usability.

LPT's claim about the independence of a component was examined in two other experiments. Two factors were studied which cause components to influence one another, i. e. consistency and mental effort. The first experiment explored the effects that inconsistency has at various layers of the user-system interaction. In the experiment, 48 university students operated PC simulations of room thermostats, web-enabled TV sets, microwaves and radio alarm clocks. The effect of inconsistency was examined between components on the same or on different layers, and between components and the application domain. The results of the experiment showed that components in a user interface could affect each other's usability significantly. Components in the same layer or in other layers can activate an inappropriate component-specific mental model, which users apply to understand the feedback of another component. The inconsistency between the application domain and a component's feedback was not found to affect the component's usability. Whether this only was the case in this experiment or can be generalised, is a topic for further research. However, the study did show that the application domain had an effect on the users' understanding of the functionality the component provides.

In another experiment it was shown that mental effort could link the control of higher-level components to lower-level components. A poor implementation of the lower-level layers can force users to adopt less efficient higher-level control strategies to cope with a task that is mentally over-demanding. In this experiment, 24 university students had to solve equations with calculators, which were composed of different versions of a component that operated on a low-level layer. One calculator was equipped with a small display, which could only display one value. The other calculator was equipped with a large display, which could display 5 lines of 34 symbols each. The subjects' cardiovascular activity was recorded as well as the message exchange between the components. Furthermore, after solving an equation, the subjects rated the effort experienced on the Rating Scale Mental Effort and, at the end of the experiment, the subjects filled out a questionnaire with ease-of-use and satisfaction questions regarding the calculators and their specific components. Results showed a significant interaction effect between the equation difficulty and the display size in the heart-rate variability, which is regarded as a mental effort index. Next, a significant interaction effect between the same variables was found in the number of times subjects

stored intermediate outcomes in a calculator, which is regarded as a change in control strategy of the high-level component.

Based on the conducted research the following answers can be given to the question: *Is usability compositional?* A first answer is *yes*; LPT indeed provides a basis for empirical evaluation of user interface components. A second answer is *no*; the usability of the entire user interface can not always be predicted solely on the usability of the individual components. Inconsistency and mental effort are factors that allow one component to reduce the users' ability to control another component. Therefore, components should not be designed, deployed and evaluated entirely independently of other components in the user interface. Within the component-based software engineering approach, the creation and deployment processes are separate. Therefore, two kinds of usability evaluations are advised: one focusing on components in isolation when they are created, and the other focusing on the component within the context of the particular application in which they are deployed.

Samenvatting (Summary in Dutch)

Dag in, dag uit worden mensen geconfronteerd met apparaten die ze moeten bedienen om allerlei redenen. De ontwerpers van deze apparaten hopen dat hun creaties gemakkelijk te bedienen zijn. Hiervoor kunnen ze zich wenden tot mens-systeem interactie theorieën die hen kunnen leiden bij het ontwerp en de analyse. Echter, tot nu toe sluiten empirische methodes om de bruikbaarheid van apparaten te analyseren gebrekkig aan bij de, steeds vaker aangewende, componentgerichte ontwikkelingsbenadering van programmatuur. In plaats dat ieder product helemaal vanaf het begin wordt ontwikkeld, worden producten in de componentgerichte ontwikkelingsbenadering geproduceerd door het samenstellen van reeds bestaande componenten (zoals een dialoogvenster, een schuifbalk, een keuzelijst, een werkbalk). De bruikbaarheid van componenten wordt momenteel niet individueel vastgesteld, maar alleen afgeleid op basis van hun invloed op de algemene bruikbaarheid van een product (zoals het aantal toetsaanslagen, taaktijd, of een vragenlijst die zich richt op het algemene gebruiksgemak en de tevredenheid).

De Layered Protocol (gelaagde protocollen) theorie (LPT) beschouwt interactie als een uitwisseling van berichten tussen componenten en de gebruiker. LPT ontleedt de mens-systeem interactie in verschillende lagen die onafhankelijk ontworpen en geanalyseerd kunnen worden. De theorie claimt de mogelijkheid van een componentspecifieke bruikbaarheidsevaluatie. Een dergelijke evaluatie zou erg welkom zijn, omdat bij de componentgerichte ontwikkelingsbenadering het ontwikkelen en het inzetten van een component plaats vindt in twee verschillende processen. Bruikbaarheidsevaluatie van een component tijdens het ontwikkelproces van de component zou efficiënter zijn dan het evalueren van zijn bruikbaarheid iedere keer dat de component gebruikt wordt in de assemblage van een nieuw product. Een bruikbaarheidsevaluatie in het assemblageproces is overbodig wanneer de algemene bruikbaarheid van een toepassing alleen afhankelijk is van de bruikbaarheid van de individuele componenten. Dit laatste is het geval volgens LPT, omdat de componenten niet beïnvloed worden wanneer componenten in onderliggende lagen vervangen worden, zolang deze componenten maar dezelfde diensten aanbieden aan de componenten in de voor hen bovenliggende lagen.

Tot nu toe is LPT alleen toegepast voor het analytisch evalueren van gebruikersinterfaces. Echter, LPT is ook naar voren geschoven als basis voor empirische evaluaties van de bruikbaarheid van individuele componenten. Van belang bij de verwezenlijking van een dergelijke evaluatiemethode is de claim omtrent de onafhankelijkheid van de componenten.

Deze was nog niet empirisch onderzocht. Het proefschrift behandelt daarom de volgende onderzoeksvraag: *Is bruikbaarheid opdeelbaar?* Deze vraag heeft de volgende onderliggende vragen:

1. Kan de bruikbaarheid van een component empirisch getest worden en zo ja, op welke manier?
2. Kan de bruikbaarheid van een component beïnvloed worden door andere componenten en zo ja, op welke manier?

Het onderzoek werd uitgevoerd in een reeks van laboratorium experimenten waarin proefpersonen taken moesten uitvoeren met verschillende op de PC gesimuleerde consumentenproducten. Het eerste experiment richtte zich op het vinden van een componentspecifieke objectieve prestatie maat. Tachtig universiteitsstudenten bedienden een fictief gebruikersinterface in dit experiment. Voorafgaand ontvingen de proefpersonen één van de acht instructiesets in een trainingssessie. De instructiesets waren samengesteld uit het verstrekken of onthouden van informatie omtrent drie componenten. Voor- en nadat de proefpersonen de taken hadden uitgevoerd is hun kennis omtrent de drie componenten getoetst. Daarnaast is gedurende de taakuitvoering de berichtuitwisseling tussen de componenten vastgelegd in een logbestand. De resultaten van het experiment laten zien dat de kennis die proefpersonen hadden over een component, het aantal berichten dat deze component ontving beïnvloedde. Dit deed vermoeden dat het aantal berichten dat een component ontvangt, beschouwd mag worden als een componentspecifieke objectieve prestatie maat. De maat geeft de inspanning aan die gebruikers moeten leveren om hun perceptie van een component te sturen. Het versturen van een bericht geeft aan dat gebruikers ontevreden zijn met de waargenomen systeemtoestand en dat zij zich inspannen deze te laten overeenstemmen met de door hun gewenste toestand.

Op grond van dit experiment is een methode ontwikkeld waarmee de bruikbaarheid van een component getoetst kan worden. De methode ondersteunt twee toetsingsparadigmata, voor een enkele versie en voor meerdere versies. In het geval van een enkele versie wordt van iedere component maar één versie getoetst. De toets richt zich op het identificeren van componenten die de algemene bruikbaarheid van een product belemmeren. In het geval van meerdere versies worden meerdere versies van een component met elkaar vergeleken. De toets richt zich hierbij op het identificeren van de meest bruikbare versie.

In het geval van een enkele versie wordt het aantal berichten dat een component ontvangt afgezet tegen de prestatie van een ideale gebruiker, daarnaast wordt het gecorrigeerd voor bedieningseffecten van componenten in de boven- en onderliggende lagen. Aan elk bericht wordt een inspanningswaarde toegekend. Deze inspanningswaarden zijn gebaseerd op de inspanningswaarde voor het versturen van berichten op het laagste niveau, die op hun beurt doorberekend zijn naar berichten op hogere niveaus. Op het laagste niveau worden er weegfactoren toegekend aan de berichten, die de inspanning om een bericht te verzenden op dit niveau moeten weergeven.

De componentspecifieke subjectieve maten voor het gebruiksgemak en de tevredenheid worden verkregen met een standaard vragenlijst. Verondersteld werd dat deze maten effectiever waren dan algemene subjectieve maten, omdat specifieke vragen de gebruiker helpen bij het herinneren van de bedieningservaringen van een specifieke component.

De methode is geëvalueerd in een tweede experiment door algemene en componentspecifieke maten met elkaar te vergelijken op hun capaciteit om bruikbaarheidsproblemen te detecteren die vooraf gecreëerd waren. Acht verschillende prototypes van een mobiele telefoon werden geconstrueerd door voor drie componenten twee versies te ontwerpen. De versies werden zo ontworpen dat ze duidelijk verschilden in hun bruikbaarheid. Tachtig universiteitsstudenten, verdeeld over acht groepen, moesten ieder een reeks taken uitvoeren met één van de prototypes. De resultaten van het experiment lieten ondanks de verwachting niet zien dat componentspecifieke subjectieve maten beter zijn dan algemene subjectieve maten in het opsporen van verschillen in het gebruiksgemak en de tevredenheid binnen het toetsingsparadigma voor meerdere versies. Dit kan echter een tekortkoming zijn van de experimentele opzet. De componentspecifieke en algemene vragen waren door elkaar gesteld, waardoor de ervaring opgeroepen door de specifieke vragen ook de beantwoording van de algemene vragen beïnvloed kan hebben.

De resultaten van het experiment lieten wel zien dat, mits de componenten onafhankelijk van elkaar zijn, een componentspecifieke objectieve prestatie maat bruikbaarheidsverschillen effectiever constateert dan een algemene objectieve prestatie maat. Dit geldt zowel voor componenten die opereren in een hoge als in een lage laag. De kracht van de componentspecifieke prestatie maat ligt in het reduceren van de statistische variantie door de analyses op één component te richten en aldus de variantie te minimaliseren die ontstaat doordat gebruikers andere componenten bedienen. Binnen het kader van het toetsingsparadigma voor een enkele versie laat het experiment zien dat de componentspecifieke objectieve prestatie maat goed correleert met algemene en componentspecifieke subjectieve bruikbaarheidsmaten en dat op basis van deze maat de componenten gesorteerd kunnen worden op hun potentie om de bruikbaarheid te verbeteren.

De claim van LPT omtrent de onafhankelijkheid van een component is onderzocht in twee andere experimenten. Twee factoren zijn bestudeerd die er voor zorgen dat componenten elkaar kunnen beïnvloeden, namelijk: consistentie en mentale inspanning. Het eerste experiment richtte zich op het effect dat inconsistentie kan hebben op verschillende lagen van de mens-systeem interactie. Achtenveertig universiteitsstudenten bedienden in een experiment verschillende PC simulaties van een kamerthermostaat, een webtelevisie, een magnetron en een wekkerradio. Het effect van inconsistentie werd bestudeerd in drie situaties, namelijk: (1) tussen componenten opererend in dezelfde laag, (2) tussen componenten opererend in verschillende lagen en (3) tussen een component en het applicatiedomein. De resultaten van het experiment laten zien dat componenten in een gebruikersinterface elkaars bruikbaarheid significant kunnen beïnvloeden. Componenten in dezelfde of in een andere laag kunnen een ongeschikt componentspecifiek mentaal model activeren dat een gebruiker toepast om de informatie van een andere component te interpreteren. Geen effect voor de inconsistentie tussen een component en het applicatiedomein werd gevonden. Of dit

alleen geldt voor dit experiment of dat dit gegeneraliseerd mag worden is een vraag die blijft openstaan voor vervolgonderzoek. Echter, het onderzoek liet wel zien dat het applicatiedomein invloed had op het gemak waarmee gebruikers de functionaliteit van de component gebruikten.

Het laatste experiment laat zien dat de mentale inspanning de bediening van een component in een hogere laag kan verbinden met de bediening van een component in een lagere laag. Een slecht ontworpen component in een lagere laag kan gebruikers dwingen minder efficiënte besturingsstrategieën aan te wenden voor een component in een hogere laag wanneer de gebruikers geconfronteerd worden met een te zware mentale taak. In het experiment losten 24 universiteitsstudenten berekeningen op met twee gesimuleerde calculators. De calculators hadden twee verschillende implementaties van een component op het laagste niveau. Eén calculator had een klein venster dat maar één getal tegelijk kon laten zien. De andere calculator had een groot venster dat 5 regels met 34 symbolen tegelijk kon laten zien. Tijdens de taakuitvoering werden van iedere proefpersoon de cardiovasculaire reacties en de berichtuitwisseling tussen de componenten geregistreerd. Daarnaast beoordeelden de proefpersonen de mentale inspanning die nodig was na het uitvoeren van iedere berekening. Aan het einde van het experiment vulden de proefpersonen ook nog een vragenlijst in over het gebruiksgemak van en hun tevredenheid over de calculators en de componenten. De resultaten van het experiment laten een significante interactie zien in de hartslagvariabiliteit tussen de moeilijkheidsgraad van een som en de grootte van het venster. Hartslagvariabiliteit geldt als een fysiologische maat voor mentale inspanning. De resultaten laten ook een significante interactie zien tussen dezelfde factoren maar dit maal in het aantal keer dat de proefpersonen tussenresultaten opsloegen in het geheugen van een calculator. Dit kan geïnterpreteerd worden als een strategieverandering voor het bedienen van een component in de hogere laag van de calculator.

Terugkijkend op het onderzoek kunnen de volgende antwoorden gegeven worden op de onderzoeksvraag: *Is bruikbaarheid opdeelbaar?* Een eerste antwoord is *ja*; LPT verschaft een kader voor empirische evaluaties van gebruikersinterface componenten. Een ander antwoord is *nee*; de bruikbaarheid van de gehele gebruikersinterface kan niet altijd correct worden voorspeld aan de hand van alleen maar de bruikbaarheid van de individuele componenten. De bediening van een component kan de bruikbaarheid van andere componenten verminderen door factoren zoals de consistentie en de mentale inspanning. Een component moet daarom niet geheel onafhankelijk van de andere componenten in de gebruikersinterface ontworpen, ingezet en geëvalueerd worden. Zoals eerder vermeld, is in de componentgerichte ontwikkelingsbenadering de ontwikkeling van een component en een product van elkaar gescheiden. Daarom wordt er voor twee soorten bruikbaarheidsevaluaties gepleit; één die zich richt op de component zelf, tijdens de ontwikkeling hiervan, en één die zich richt op de component binnen de context van een bepaald product, wanneer de component daadwerkelijk wordt ingezet.

Curriculum vitae

- 30 Dec. 1970 Born in Middelburg, The Netherlands
- 1984 - 1988 Klarenbeek LTS (junior technical school), Middelburg
— Installation Technology
- 1988 - 1991 MTS (intermediate technical school), Zeeland College, Vlissingen
— Information Technology
- 1991 - 1995 Hogeschool Eindhoven (technical college)
— B. Sc. Information Technology
- 1995 - 1998 Technische Universiteit Eindhoven (university of technology)
— M. Sc. (cum laude) Technology & Society
- 1996 - 1997 Part-time position at the usability laboratory of the Rabobank
- 1998 - 2003 Technische Universiteit Eindhoven (university of technology)
— Ph. D. at the J. F. Schouten School for User-System Interaction
Research