

Linköping Studies in Science and Technology

Thesis No 576

Commenting Systems as Design Support

A Wizard-of-Oz Study

Mikael Ericsson



Submitted to the School of Engineering at Linköping University in partial fulfillment of the requirements for the degree of Licentiate of Philosophy

Department of Computer and Information Science
S-581 83 Linköping, Sweden

Linköping 1996

Commenting Systems as Design Support

A Wizard-of-Oz Study

by

Mikael Ericsson

October 1996

ISBN 91-7871-833-3

Linköping Studies in Science and Technology

ISSN 0280-7971

Thesis No 576

LiU-Tek-Lic 1996:41

ABSTRACT

User-interface design is an activity with high knowledge requirements, as evidenced by scientific studies, professional practice, and the amounts of textbooks and courses on the subject. A concrete example of the professional need for design knowledge is the increasing tendency of customers in industrial systems development to require style-guide compliance. The use of knowledge-based tools, capable of generating comments on an evolving design, is seen as a promising approach to providing user-interface designers with some of the knowledge they need in their work. However, there is a lack of empirical explorations of the idea.

We have conducted a Wizard-of-Oz study in which the usefulness of a commenting tool integrated in a design environment was investigated. The usefulness measure was based on the user's perception of the tool. In addition, the appropriateness of different commenting strategies was studied: presentation form (declarative or imperative) and delivery timing (active or passive).

The results show that a commenting tool is seen as disturbing but useful. Comparisons of different strategies show that comments from an active tool risk being overlooked, and that comments pointing out ways of overcoming identified design problems are the easiest to understand. The results and conclusions are used to provide guidance for future research as well as tool development.

This work has been support by the Swedish Council for Research in the Humanities and Social Sciences.

Department of Computer and Information Science
Linköping University
S-581 83 Linköping
Sweden

Acknowledgments

My colleagues, friends and family have been providing invaluable help and support during the making and completion of this thesis. I am indebted to:

Supervisors and project members


Jonas Löwgren, my supervisor, for daily co-operation, guidance, discussion, criticism, encouragement and motivation,

Magnus Baurén, and Yvonne Wærn for collaboration and discussions during the CODEK project,

Sture Hägglund and Ulf Nilsson, co-supervisors, for comments and criticism.

The experiment participants, for making this study possible.

Colleagues, friends and family

The rest of the members in the -group—Pelle, Torbjörn, Stefan—thanks for your time, and especially for creating a creative and interesting research and education environment,

Joachim Karlsson, Niclas Ohlsson, Göran Forslund and other colleagues in the Application Systems Laboratory, for the exchange of ideas, criticism, comments and support during our parallel journeys towards our theses,

Andreas Björklind, Patrick Doherty and other friends and colleagues at the Department of Computer and Information Science—not for making things easier, but much more fun!

Finally, I want to thank my family for their support and encouragement.

Mikael Ericsson

Linköping, August, 1996

This work was financially supported by the Swedish Research Council for Social Sciences and the Humanities (HSFR).

TeleUSE is a registered trademark of TeleVerket, Inc. OSF Motif, and OSF/Motif are trademarks of Open Software Foundation, Inc. UNIX is a registered trademark of AT&T Bell Laboratories, Inc. Open Windows and Developer's Guide are trademarks of Sun Microsystems, Inc. SPARC is a registered trademark of SPARC International, Inc. Apple Macintosh is a registered trademark of Apple Computers, Inc. Microsoft Windows is a registered trademark of Microsoft Corporation. Netscape Navigator is a registered trademark of Netscape Communications Corporation.

Contents

1 INTRODUCTION	1
Research questions	3
Aim and rationale	5
Contributions	7
Overview of the thesis	8
2 USER INTERFACE DESIGN SUPPORT TOOLS AND COMMENTING SYSTEMS	11
Design in software projects	12
User interface design knowledge and information	13
Guidelines	16
Supporting the use of guidelines	23
Commenting Tools as DSS	29
Evaluation and commenting support tools for user interface design—some illustrations	40
3 RESEARCH APPROACH AND EXPERIMENT DESIGN	45
Objects to think with	47
Design and evaluation of design support tools	47
Background study	51
Using the Wizard-of-Oz approach	53
Clarifying the research questions	55
Experiment design	64
Pilots	69
4 THE CODEK SOFTWARE ENVIRONMENT	71
Simulating a design environment with a commenting component	72
The simulated environment	74
Low-level issues	84
5 THE EMPIRICAL STUDY	89
Experimental design	89
Written material	90
The CODEK experiment environment	92
The Wizard and the guidelines	95
Procedure	100
Collecting, coding and analysing data	104

6 RESULTS	109
Wizard performance	109
Participant perception of task	111
Usefulness of a commenting system	112
Effects of different commenting strategies	113
Transfer of knowledge/behaviour	116
Perceived MWL	118
Interpretations of the interaction logs	119
Participant comments and interview summary	120
7 CONCLUSIONS	123
Usefulness of commenting tools	123
Appropriateness of comments	123
Perceived MWL and comment detection	124
Knowledge-transfer effects of using a commenting tool	125
Correlation between activity measures and subjective situation ratings	126
Design recommendations for commenting tools	128
8 DISCUSSION AND FUTURE WORK	131
Applicability of the results	132
Methodological issues	137
General discussion	144
Future work	146
REFERENCES	153
A DESIGN-TASK MATERIAL	169
Written task specification	169
B QUESTIONNAIRES	171
Questions about the design environment	171
Questions about the commenting tool (general)	171
Questions about a specific situation	172
C PERCEIVED MWL MEASUREMENT FORMS	175
TLX form	175
D EVALUATION-TASK MATERIAL	177
The experts' written instructions	177
The interface proposals	178
E SESSION INTERACTION DIAGRAMS	181
Contents of Interaction Diagrams	181
Interaction Diagrams 9, 14 and 16	181

Chapter 1

Introduction

Designing software is a knowledge-intensive activity, and the process of designing and implementing the user interface (UI) of a system is no exception. Being involved in the user interface design activities often means taking on a set of different roles during a software development project, e.g., the systematic engineer, the creative artist or the systems analyst. All these roles demand large amounts of information, which are not only required to be accessible, but also integrated, during work. The amount and complexity of the information is likely to generate problems during the design work. There is a need to support the designer in finding and handling relevant information.

An important part of current user interface design work is the use of guidelines, i.e., collections of (written) advice and rules meant to support the designer. Guidelines may serve many different purposes. In some situations, the guidelines may be intended as a general design support tool, merely packaging design knowledge relevant for a specific project. In other situations, the intention might be to force the designers¹ to conform to a specific “design style”, as is the case of style guides. There is also a growing tendency to include requirements for style guide or standards compliance in product specifications, for strategy reasons or in order to conform to major standards, national regulations or even laws. However, studies of designers using guidelines show that current printed material suffer from several problems, e.g., they are difficult to interpret and use

¹ Throughout this thesis, we will use the term “designer” in a rather broad sense; the general meaning will be a “software designer that is involved in user interface design work”.

(Thovtrup and Nielsen, 1991; de Souza and Bevan, 1990; Hammond et al., 1987; Tetzlaff and Schwartz, 1991). Providing the designer with useful tools for handling this kind of information becomes increasingly important.

Current user interface design tools provide the designers with support for sketching, prototyping, implementing, redesigning and reusing interfaces and interface components; in other words, support for creation and modification. To our best knowledge, there are no commercial tools supporting the delicate task of integrating guideline knowledge in the design process. Still, the inclusion of evaluative power and guideline knowledge is rated important in the literature (e.g., Perlman, 1988; Hartson and Boehm-Davis, 1993; Sweeney et al., 1993; Keller, 1994).

We believe that the designer would benefit from the integration of *useful* tools supporting design information management, and guideline knowledge in particular. In recent human-computer interaction (HCI) research, there have been several attempts to create tools for handling the different kinds of knowledge in the designer's environment. Ranging from techniques for textual representations of design knowledge to automated evaluation tools, several research prototypes have been built. However, very few of those have been evaluated from the perspective of usefulness.

One technical approach to integrate information and knowledge support in the designer's environment is to use a *commenting* or *critiquing* component. This can be described as a sort of agent that monitors the designer's work (the interface under development) in the design environment and delivers comments aimed at supporting her (Silverman, 1992a; Hägglund, 1993). The comments may originate from, e.g., guidelines and style guides implemented as rules, or examples and case-based reasoning. Although many commenting prototypes have been built, most projects have focused on technical features and performance rather than usefulness and actual impact on the process and product. Before including such tools in professional environments, they should be evaluated: How are they perceived by the user (the designer)? How is the product (the interface developed) affected? Is this an effective support approach? What is the impact on the design process and future projects?

Our work is aimed at exploring in more detail the usefulness of and requirements on an evaluative design support system. We concentrate on systems that provides the designer with knowledge relevant for the design work, by commenting on a design solution, in the context of the design task at hand. In this thesis we focus on the perceived usefulness and the

effects of different commenting strategies. We also investigate the potential knowledge transfer effects of using a commenting support tool and possibilities to improve commenting strategies based on the designer's interaction.

In order to approach these questions, we decided to perform a small-scale empirical study of potential designers using a design environment that included a commenting component. Of course, this didn't allow us to investigate real workplace and process impact, but, as we explain later on, the data collected here is most important *before* conducting experiments in a professional context. In the study, including 16 participants, we used so called Wizard-of-Oz (WOz) techniques to collect data. In short, the idea is to let the participants believe that they use a real system, but instead let a human operator (the Wizard) simulate part of the functionality. This makes it possible to test and explore new concepts and techniques without actually fully implementing them. This kind of method has been used in several other domains, e.g., natural language communication analysis and usability evaluation (Fraser and Gilbert, 1991; Coutaz et al., 1993a), but has not been used to a great extent when developing new design support tools.

The empirical study presented here, investigating the issue of communicating design knowledge, will be referred to as the CODEK (COmmunication of DEsign Knowledge) project.

1.1 Research questions

The goal of the CODEK project was to study the effects of using commenting knowledge-based (KB) tools for user interface design support. This goal is threefold, consisting of sub-goals related to overall *usefulness*, optimal system *behaviour* (strategy) in terms of delivering comments, and potential *learning effects*. More specifically, the research questions addressed in this work are:

1.1.1 Is a commenting tool useful in user interface design work?

Our main focus of interest is the usefulness of a commenting design support tool. The usefulness of such a tool might be defined in many different ways, including measures of productivity, quality, etc. In our study, we evaluate the system from the point of usefulness as perceived by the designer. We also explore and discuss it in the perspective of transfer of

design knowledge, and the number and severity of design flaws eliminated by using the tool.

1.1.2 What is the most appropriate way to deliver comments?

Given that a commenting tool is rated useful, what is the most appropriate way to deliver the comments? In order to gain knowledge about the advantages and disadvantages of different strategies we study the form of the comments and the behaviour of the tool.

What is the most appropriate form of the comments?

When delivering comments, the actual comment may be composed and presented in several different ways: as text, with or without a rationale, using visual cues, etc. In our study we distinguish between *declarative* comments, that merely point out the existence of a possible design flaw, and *imperative* comments, that also suggest how the flaw can be remedied. This dimension will be referred to as the *mood* of the comments, using a concept borrowed from linguistics. We investigate the effects of using different moods when presenting comments.

What is the most appropriate behaviour of the commenting tool?

Commenting systems may use different strategies, or behaviour, to deliver comments. We focus on the distinction between *passive* and *active* tools. A passive tool requires the designer to ask explicitly for comments. An active tool autonomously presents a comment as soon as it detects a design flaw. This dimension is called the *mode* of the commenting tool. In this study, we compare the effects of using different modes for commenting.

1.1.3 Does using a commenting tool imply knowledge transfer?

As mentioned earlier, the usefulness of a design support tool might be defined in many ways.

From a quality and effectiveness perspective, it would be valuable if a commenting tool had learning effects, i.e. if the designer would gain knowledge from using the tool. This would integrate the learning in the design process, possibly reducing training costs and making it easier to relate guidelines to their targets.

For these reasons, we want to explore potential knowledge transfer from the use of a commenting tool.

1.1.4 Other issues of interest

While dealing with questions concerning different strategies for commenting tools it becomes necessary to deal with system control and algorithms—what kind of algorithm should be used to control the tool behaviour? This issue is explored by relating the user's perception of the design work (e.g., in terms of perceived mental workload (MWL)) to the interaction (in terms of actions in the design tools). The informal question is: Is it possible to improve (in terms of e.g. the designer's perception of the system) the comment delivery timing using low-level interaction data?

We also touch upon some issues of relevance for continued studies of design support tools—advantages and disadvantages of using Wizard-of-Oz techniques in this domain.

1.2 Aim and rationale

Our research has been guided by two interrelated aims: to support the user interface designer by making design knowledge easier to access and to improve today's design support tools. In addition, we also wanted to explore the Wizard-of-Oz technique as a methodological approach when addressing that kind of questions.

1.2.1 Current technical support is not adequate

When it comes to the design and actual implementation of user interfaces, the designer has access to tools that support prototyping, layout, callback-generation, etc., all of which aid creation and implementation. However, almost no parts of current tools help the designer to access or make use of the vast amount of information and knowledge relevant for the design process: customer specification, user input, general guidelines, company style guides, industry standards, etc. Perlman (1988) proposes user interface design tools that incorporate design knowledge originating from experience and research. That way, he argues, the knowledge can be transferred to current and future system development projects more economically than using other methods such as textbooks or employee migration.

By adding on-line documentation or hypertext systems to the designer's environment (Perlman, 1989, Ogawa, 1993, Philips, 1993), the designer is given much more information, but the main problem remains;

an overflow of information is available but not prepared for actually supporting the designer. It is *included*, but not *integrated* in the designer's environment.

A related motivation for this study is the lack of evaluative power in today's user interface tools (Perlman, 1988; Hartson and Boehm-Davis, 1993; Keller, 1994; Myers, 1994). There exist systems that aid usage logging and usage pattern analysis, but few tools address evaluation aspects during the design and implementation. By integrating systems that provide relevant design knowledge, e.g., in the form of guidelines or style guides, it might be easier to avoid design errors earlier in the process, thereby reducing the need for later, and often expensive, redesign.

In addition, standards are an emergent factor in current systems development. Style guide compliance is often a competitive issue, and sometimes standard conformance is a requirement from the customer. In the European Community, the directive EC 90/270 (EEC, 1990) requires employers to inform and protect workers of visual display screen equipment and to ensure compliance regarding certain man-machine interface issues. This directive is now becoming national laws in the EC countries. Meanwhile, several international standards concerning HCI are being produced. ISO 9241 (ISO, 1992) specifies ergonomic requirements for office work with visual display terminals, and is used as a basis for many of the national regulations following the EC directive. Nine of the seventeen parts of the 9241 document address issues like usability, task requirements, information presentation, user guidance and dialogue principles.

The increasing demand for using guidelines and style guides adds further to the request for new tools, as reported in previous research. In a study of designers' requirements on design support tools, Rosson et al. (1988) found a consensus regarding the importance of providing support for handling external business requirements, e.g., style guide compliance. Several studies report on the difficulties of using standard, printed, guideline documents (Hammond et al., 1987; de Souza and Bevan, 1990; Thovtrup and Nielsen, 1991; Tetzlaff and Schwartz, 1991), and suggest alternative approaches, e.g., using knowledge-based techniques.

In the area of knowledge-based design support systems, previous research has demonstrated different technical solutions to different parts of the information problem (e.g., Löwgren and Nordqvist, 1992, Weitzman, 1992, Harstad, 1993, Fischer et al., 1993a). Instead of just presenting raw information, relevant knowledge is prepared and displayed for the task at hand, throughout the process. In commenting, or critiquing,

support tools, the information is delivered by commenting (or critiquing) the designer's current work (Silverman, 1992a; Hägglund, 1993).

1.2.2 Empirical studies are needed

A strong motivation for carrying out this study was the prior lack of empirical results concerning commenting support tools. We argue that there is a need to evaluate those techniques now, and to ascertain the usefulness of this kind of tools, before integrating them in a professional setting (or rejecting them). We do not claim to cover this in full, but only report a first step in that direction.

Several research prototypes have been built to prove the technical feasibility of such tools (see above and section 2.6), and in the area of knowledge-based systems, critiquing systems has been dealt with for more than a decade (Hägglund, 1993).

Some research reports exploratory studies or informal data, and in some cases also propose guidelines for future research (Carroll and Aaronson, 1988; Lemke and Fischer, 1990; Silverman and Mazher, 1992; Silverman, 1992a; Löwgren and Laurén, 1993; Bonnardel and Sumner, 1994), but so far, there are few reported empirical studies in the area of commenting user interface design support tools. Our work contributes some results and possibly presents new questions.

1.2.3 Wizard-of-Oz—a viable approach?

A third motivational component has been the method used to collect the data—the Wizard-of-Oz technique (Dahlbäck et al., 1993). This method has been used in other domains (e.g., natural language communication, see section 3.4) to collect data; we wanted to explore the method in the context of graphical user interface design tools. Making this study possible, the technique at the same time presents some delicate questions about validity and reliability. The data and experience from our experiments provided some material for analysis and discussion of these matters.

1.3 Contributions

The research described in this thesis has resulted in three major contributions. First, the main part of this thesis is about an empirical study of commenting tools for user interface design support. The results from the experiment activities answer questions about the usefulness of such tools, and address questions about commenting strategies and the potential edu-

cational value of commenting tools. Based on the results from our empirical study, we conclude that:

- a commenting tool is considered useful by designers when working with user interface design tasks.
- users of a commenting tool prefer receiving comments indicating suggested improvement, instead of mere information about detected problems.
- using a commenting tool affects the designer's evaluation behaviour, i.e., there is some indication that knowledge transfer occurs.
- a high perceived mental workload (MWL) relates to problems detecting comments when using a commenting tool.

Secondly, we present some tentative design recommendations for future work with commenting tools.

Thirdly, in order to perform the experimental study, a Wizard-of-Oz environment for studying user interface design support tools was designed and implemented. Hopefully, the technical description of the system can be used as input by others wanting to use the same method. The Wizard-of-Oz approach is a non-trivial method, with potential problems related to validity, reliability and ethical issues in research. A large part of this thesis can be described as a report of our experiences from using Wizard-of-Oz techniques to study user interface design support tools. We address the problem areas along the presentation of our work and describe our approaches in critical situations.

1.4 Overview of the thesis

This thesis is organized with the aim to make it valuable for at least two groups of people: those who work with design support tools and commenting systems, and those who are primarily interested in Wizard-of-Oz based techniques for evaluation or research. The reader is advised to read the chapters in order.

Introduction

The first chapter gives the background, settings and aims of this thesis. We present the research questions, our rationale for carrying out this research project, and our major contributions.

User interface design support tools (and commenting systems)

The second chapter gives a framework for the concepts, techniques and systems discussed throughout the thesis. The purpose of the chap-

ter is twofold. First, we want to give a context for the use of user interface design guidelines and user interface support tools, and secondly, we want to explain the concept of commenting support tools, and to describe the environment in which they exist. This chapter is also used to frame our research problem.

Research approach and experiment design

After having presented the framework for our work, we present our research approach. In the third chapter we describe methodological aspects of our work, and we explain our experiment design. The purpose of this chapter is to make it possible to reflect on issues related to validity and reliability in the succeeding chapters. We also present our view on the value of the Wizard-of-Oz method in this kind of research.

The CODEK Software Environment

The fourth chapter presents the implementation-related parts of our work: the Wizard-of-Oz environment used in the experiments. The rationales for design solutions in the environment are presented. This chapter makes it possible to interpret our results, in a technical perspective, and also presents ideas for future research using Wizard-of-Oz techniques.

The empirical study

In the fifth chapter, we present the primary part of our research: the empirical study of commenting support tools for user interface design. This chapter contains information about the actual settings, tasks, material, participants and the procedure used.

Results

The sixth chapter presents the results of the empirical study.

Conclusions

In the seventh chapter we move on to presenting our conclusions from the study. In addition to addressing our research questions, we also draw conclusions of importance to future studies of design support tools and the actual design of such tools.

Discussion

Finally, we discuss the implications of the conclusions. This chapter is divided into three parts. First, we discuss the applicability and implications of our results. Secondly, we raise and try to answer some questions related to validity and reliability, but also ethics in the kind of

empirical research conducted in this study. The chapter ends with a discussion about directions for future research.

Appendices

A set of appendices presents the material used in the actual empirical study: questionnaires, task specifications, etc.

Chapter 2

User Interface Design Support Tools and Commenting Systems

The design of a software product can be described in many different ways, depending on the perspective taken by the viewer. In reality, design is a multitude of activities (Bellotti, 1988; Edmonds, 1994; Myers, 1994), requiring several different skills, often carried out by the same people in different roles. In some situations it is mostly creative or artistic, in others it is largely a task for an engineer. Part of the work consists of analytic reasoning while other requires the ability to search completely new solutions, free from constraints, in an open problem space.

The word “design” has several connotations, even in software design. It is important to recognize this in order to develop tools and techniques that support the designer. Before we can address our research questions correctly, we have to explore the environment to which they belong. We need to understand the tasks and problems of the designers. Hence, we need to identify the most important features of user interface design.

We also need to clarify the terminology, since many of the terms used exist in several different domains related to our work. From here on, we will use the term “design work” to describe the situation where one designer works on a user interface design problem. We focus on the phases related to detailed design, i.e., after conceptual design and before evaluation. We are also mostly interested in the parts where the work is carried out primarily with the help of computer-based tools, e.g., hi-fi prototyping², interface layout, interface implementation, etc.

2.1 Design in software projects

What is the nature of a *design task* or a *design problem* in software projects?

Design problems have previously been described as ill-defined, open-ended or “wicked” (Simon, 1981; Rittel and Webber, 1984). In the user interface domain, this notion has been interpreted as (Fischer et al., 1993b; Bonnardel and Sumner, 1994; see also Löwgren, 1995):

- Design problems are complex in terms of the amount and content of the information that has to be available to and handled by the designer.
- There is no single, best solution for the task at hand.
- In order to gather information relevant for the problem, one has to understand the problem, but the problem cannot be understood without information about it.

If we accept this interpretation of design problems, we immediately see some of the sources of potential problems: 1) information overflow and complexity, 2) an inherent need for experience and heuristics, and 3) a large portion of creativity and analytic ability in combination.

Design work is often described as a kind of cyclic or iterative process, in which the designer goes through the components reflection and action until the process is considered finished (Schön, 1983). In user interface design, this corresponds to a successive refinement of reflection and redesign; the designer implements part of the design, looks at it and reacts, performs adjustments or redesigns, and so on.

Of course, the process contains several other activities, but the reflection-action cycle is significant; in this we see other interpretations of potential sources of problems: it might be difficult to reflect on the current design because 1) it is hard to combine all the (complex) information relevant for the reflection, and 2) it is difficult to interpret the observations and use them for redesign. The designer may avoid the reflection, because of the difficulties. It might also be the case that much of the important information is located in external documents, which may require too much time and resources to access.

All this together put special demands on the systems needed for supporting design work. In order to aid the designer throughout the whole

² Prototyping is often divided into lo-fi (using sketches, paper mock-ups, etc.) and hi-fi (using computer-based mock-ups, authoring tools and almost complete implementations).

process, they must contain tools for several different roles: the artist as well as the engineer.

2.2 User interface design knowledge and information

What are the most important aspects of user interface design support?

Many factors influence the designer's work and contribute to the complexity of the design task. These factors may be low-level or high-level, internal or external and affect the designer implicitly or explicitly. Some of the important contributions originate from:

- the designer's experience, competence and creativity.
- the product specification, more or less formalized.
- data and specifications from previous phases: task analysis, prototyping, etc.
- specifications and guidelines from the design team.
- the user interface tools used for the actual implementation (limitations, technical capabilities).
- external demands from customers, national laws, industry standards, etc.

Some of the factors are "hard" in that the associated specifications or rules must be followed; the designer has no choice (e.g. product specification and national laws). Others are "soft", meaning that the information is only meant as advice or decision support; the designer decides what to do (e.g. industry standards, general guidelines). A major task of being a designer is to integrate all this information and make use of it during the design process.

When trying to categorize the factors in the perspective of the design process, many different aspects and dimensions may be used. In order to narrow the scope and explain support tool requirements we present an informal description of the nature of the information, and the problems it may cause. In this, we focus on the following entities: the *goal* of the information, the *time* at which it is presented, the *mediator* of the information, the *form* of the information and its *origin*.

2.2.1 The goal of the information

Information that is actively "inserted" into the design process often aims at increasing or improving aspects of one of three components: the resulting

artifact, the designer or the techniques used. The sub-goal may therefore be, e.g.:

- the artifact's quality, usability or plain conformance with standards or specifications.
- the designer's creativity, productivity or ability to apply standards or specifications.
- the effectiveness, flexibility or accuracy of any of the techniques used in the process, e.g., software tools or working method.

Looking at these dimensions only, it is clear that some of the information encourages creativity and flexibility, while some of it is constraining or limiting. Often, the integration of all this results in clashing or incompatible material. This, and the skills required to handle the situation, is part of the very core of design work.

2.2.2 Process phase and current task

As mentioned above, the design process consists of many different phases, not necessarily performed strictly in sequence. However, we can identify a set of sub-tasks that differ significantly from each other.

Early in a project, the designer may be involved in conceptual design and high-level modelling. At this stage, the concepts, tasks and aims are the focal points, rather than the layout, functions and code. This is a creative rather than an evaluative task.

Prototyping may be used for several different reasons: to extract more information from the users; to test several alternative design solutions; to create a common ground for discussions with the customer, etc. When creating prototypes, the designer becomes involved with lower-level issues of the task, task-to-function mapping and visual layout. Although primarily creative, this phase also includes more evaluation and comparison of alternatives.

When the visual layout and actual implementation takes place, the designer is more likely to focus on technical matters: widgets, visual layout, toolkit components, etc. At this stage, creativity and evaluation may have equal influence: on the one hand, the design specification and layout solutions from earlier phases are to be followed; on the other hand, the final layout may not yet have been decided. This kind of design work is often called detail design.

Later during the project, a more formalized phase of evaluation may take place. In this, the designer may use usability goals from earlier

phases, design heuristics and guidelines, parts of the specification or design checklists in order to evaluate the resulting artifact. The purpose of the evaluation may be to generate input for a new iteration, to validate that customer requirements are fulfilled, to measure the usability of the product, etc.

The combination of activities in the tasks—formal specification, creation and evaluation—makes the process complex. The designer has to be able to switch between creation and reflection, design and evaluation, analysis and implementation. Since the tasks described are often intertwined in a design project, at least to some degree, the complexity of the information input increases further.

2.2.3 The mediator of the information

To reach the goal, the originator may choose different persons or components as mediators for the information. Some information may be transferred between projects by means of the experience the designers gain during work. In other cases, formal training or lectures may be used to communicate the information. A third way of actively transferring knowledge is to rely on written material, delivered to the designers.

Another way to input information in the design process is to use tools or techniques—to let the method or software tools involved contain and communicate the information via affordances or constraints. For example, the method may require the designer to follow a certain standard throughout the work, or the layout tool may be locally adapted for the layout style required by the customer (e.g., to provide only a subset of all the widgets available in the window system).

Regardless of the mediator used, it is a difficult task to transfer knowledge effectively and correctly. Written material has the disadvantage that it requires a great deal of activity from the designer; it may be hard to find the time to study the documents, and even harder to find motivation for the effort. Lectures and seminars may be effective, but also requires time and even more motivation; training becomes useless if the students do not want to learn. Relying on experience or tacit knowledge can be insufficient from a performance perspective. If the technique or tool approach is used, there is a risk that the procedures and behaviour are learned by rote, but that the knowledge and actual skills are not transferred.

2.2.4 The origin and form of the information

The information may origin from sources outside as well as inside the current project. It may be formal specification information from the customer, design guidelines or rules from the design team, experiential knowledge from earlier projects in the company, usability or quality heuristics and guidelines from external sources.

As indicated above, the information may be “hard” or “soft”, presented as rules, requirements or advice. This adds further to the list of potential problems, since it is hard to find the right combination of advice and rule when aiming to transfer knowledge. If the material is presented as raw information, or advice, rather than rules and specifications, the designer may choose not to conform. If, on the other hand, the knowledge is presented as rules, made mandatory to follow, the risk of mechanical behaviour increases, as in the technique-mediator case above.

From this informal and very brief attempt to describe some of the components of user interface design, it is possible to immediately identify several potential sources of problems. We also see that one of the common factors of all the issues touched upon above is information; overflow or lack of, inadequate or too complex, too general or too specific.

There are many different types of tools aimed at supporting the designer, for several of the different sub-tasks in a design project. In this thesis we focus on the tools used for the user interface implementation and evaluation parts. More specifically, we are interested in tools aimed at helping the designers access and use information that can help improve the result of their work during detail design.

2.3 Guidelines

Guidelines are advice or rules, often in printed form, aimed at delivering design knowledge to the designer. We use the term “guideline” to describe a large variety of documents containing design knowledge and information. This general view is separated into several categories of information in subsequent sections. To exemplify the guideline concept, consider the following examples of publicly available documents:

- “Guidelines for designing user interface software” by Smith and Mosier (1986)
- “Human-Computer Interface Design Guidelines” by Brown (1988)
- “Motif Style guide”, from OSF (1991)

- “A corpus of selection rules for choosing interaction objects” by Vanderdonckt (1993)
- “ISO 9241: Ergonomic requirements for office work with visual display terminals (VDTs)” from ISO (1992)

In addition, we also consider local documents of advice or rules, based on in-house experience, standards or system specifications/requirements to be guidelines. Figure 2.1 contains some examples of guideline contents.

Applications that use data files should include a File menu, which provide all the commands the user needs to open, create, and save files.

“Windows Style guide” (Microsoft Corporation, 1992)

Assist associations between items by placing them within 5 degrees’ visual angle of each other.

“Screen design” (Tullis, 1988)

Strive for consistency

“Designing the User Interface” (Shneiderman, 1992)

Each sub-window *must* have a “Close”-button, that can be used to close the window. This button *should* not be the default action.

*Example of a local project guideline,
possibly originating from the specification*

Figure 2.1 Some examples of the contents of a guideline collection

All these document are in a way externalizations of design experience and knowledge. In order to identify potential advantages and disadvantages, we present an informal categorization of guideline aspects, similar to the one of design information in section 2.2. We use the following aspect dimensions: the purpose of the guideline, the origin and form of the material and the perspective of the content.

2.3.1 Purpose and content perspective

As indicated above, guidelines is sometimes used in order to form or try to “control” the design; to evaluate it during the work, or to validate the result afterwards. There are many different ways of looking at the “purpose” of guideline documents, but it is easy to identify at least the same categories of goals as in the description of the design information in section 2.2:

- product attributes: the artifacts' quality, usability or plain conformance with standards or specifications.
- designer education: the designers' creativity, productivity or ability to apply standards or specifications.
- process improvement: the effectiveness, flexibility or accuracy of any of the techniques used in the process, e.g., software tools or working method.

The goals are closely related to the content perspective; the view that is presented for the reader. Depending on which of the above goals that is most important, the object in focus may be any of: the designer, the resulting system, the end-user, the customer, the process, etc.

2.3.2 Origin and form

The origin of the information contained in guideline documents often affects the form in which the material is presented. A great deal of the publicly available guidelines are primarily based on either empirically based information, or plain common sense. In this informal description we distinguish between five different forms/origins:

- scientific knowledge; information based on empirical studies or psychological theory, e.g., cognitive psychology, organizational psychology. This kind of documents often provide low-level advice related to perception and psychology.
- experience; information based on practical experience, skills and common sense from previous design projects.
- established techniques; information based on *ad hoc* solutions, common use or mainstream activity.
- style guides; information selected from other sources or created from scratch, adapted for a specific "style", e.g., a certain platform, company identity or application look.
- standards; local, national or international regulations and rules that aim at ensuring quality, compatibility, safety, etc.

Another way of looking at the form of the guidelines is to focus on the *level* of the material; one extreme is the conceptual information, the other one is the low-level implementation details. This dimension is referred to as (*linguistic*) *level* (Marcus, 1991, Vanderdonckt, 1995, Bodart and Vanderdonckt, 1995), and may be described as consisting of, at least, the following different layers:

- lexical: information about physical/visual layout, interaction object attributes, etc.
- syntactic: information about dialogue structure and workflow.
- semantic: information about task-function mapping and domain related issues.

With this set of significant aspects of guidelines, we turn to a more differentiated view on various types of guideline documents.

2.3.3 Classes of guidelines

User interface design guidelines are often divided into three main types (e.g., Preece et al., 1994): high-level principles, design rules and standards. These types differ primarily in terms of their generality, origin, applicability and purpose.

High-level principles

High-level principles are general advice, with the main aim to inform and guide the design work on a framework level. Principles are expected to help integrate ideas and different parts of the work into a sound aggregate.

The principles often origin from psychological theory and/or experience. They are results of formal experiments in controlled environments, accumulated theories or formalized knowledge from previous work. This kind of guidelines are often universally applicable and must be interpreted in the current context, i.e. adapted for the task at hand. They are often presented in a way that is considered rather vague and hard to apply.

Offer informative feedback.

“Designing the User Interface” (Shneiderman, 1992)

Speak the user’s language.

“Teaching User Interface Design Based on Usability Engineering” (Nielsen and Molich, 1989)

For item differentiation, use a maximum of five colours (plus or minus two) to match the user’s short term memory capacity.

“Graphic Design for Electronic Documents and User Interfaces” (Marcus, 1991)

Figure 2.2 Examples of high-level principles.

Design rules

Design rules are specific rules, or even low-level instructions. The main purpose is to guide and instruct in the daily design work, and to direct details of a design, for the particular system in question.

Rules are often developed by platform manufacturers or local human factors groups. They are not always grounded in theory or scientific knowledge; usually they origin from experience, established techniques and common sense. Applying design rules is in most cases a simple task in comparison to the high-level principles. Because of the origin and purpose of the rules, they become project specific and more likely locally applicable.

Each menu should contain an underlined mnemonic access character that gives direct access through the keyboard.

“Windows Style guide” (Microsoft Corporation, 1992)

When closing a window, the user must be able to choose whether to save any changes made to the document since the last time it was opened.

“Apple Human Interface Guidelines” (Apple Computer, 1987)

Figure 2.3 Examples of design rules

Standards

User interface standards are prescriptions, either general or specific, with the primary goal of helping designers achieve certain kinds of consistency. Whereas principles and rules are often introduced voluntarily by the designers' company, the use of standards is often a result of customer requirements.

Standards are developed and published by local, national or international standardization organizations. In many cases they derive from previous design principles and rules; the advice and hints have been formalized. Many standards are meant to be used in different projects, by different companies, on a national or even international level. Therefore they are often presented as high-level, widely applicable rules that must be interpreted in and adapted for the current context.

The tools available in a dialogue should allow the user to adapt them to suit recurrent tasks.

ISO 9241, part 10 (ISO, 1992)

Activate the Help function automatically (or offer Help) when the user is making repeated errors.

“Human-Computer Interface Guidelines” (NASA, 1992)

Figure 2.4 Examples of standards

2.3.4 Problems with guidelines

Although aimed at supporting the design process in different ways, guidelines has been shown to cause problems for the designer., i.e. ironically, their degree of usability or usefulness tend to be rather low in some cases. In previous studies, three categories of problems have been investigated and reported. These problems are related to the access, interpretation and contextualization of the guidelines.

Access

The major reason for the problems with guidelines lies in the very nature of their representation; they are yet another piece of information that is made available, in textual, printed form, but not really integrated in the design process. Looking at traditional guidelines and their use in a design project, we identify several potential sources of problems:

- When preparing documents for the project, there is a very large amount of publicly available guideline documents to choose from, and each one often contains an immense number of guidelines.
- In one design project, the guidelines that are to be used may be spread out over several different documents and books.
- In each document, only some guidelines are meant to be used in the project.
- The guideline documents are not adapted for the needs of the current project, designer or situation.

All this may result in difficulties in accessing the guidelines. Mosier and Smith (Mosier and Smith, 1986, Smith, 1988) studied 130 users of guideline documents and found that the average time required to find specific information (one guideline) was 14 minutes. Smith (1988) also report that

only about half of the guideline users succeeded in finding guidelines relevant to their problem.

Interpretation and Contextualization

One of the basic ideas behind guidelines is to transfer knowledge from one design situation to another; to learn from experience and previous attempts. This implicitly causes problems for the designer, since the content must be presented in a form that may be used and reused in different situations and projects. This means that each guideline must be interpreted and contextualized; adapted to the task at hand.

Problems of interpretation have been found in studies by e.g., Thovtrup and Nielsen (1991) and de Souza and Bevan (1990). Tetzlaff and Schwartz (1991) found that the participants had problems interpreting the guidelines, and relied on pictorial examples, in some cases to the extent of excluding the associated text.

Another problem occurs when the designers are not experienced in human factors. Many guidelines are presented in a very brief form, without underlying rationale or explanations. Therefore, the designers find it hard to appreciate or apply the guideline information, even if it is made available to them. This situation becomes even more complex because of the many different stakeholders and participants in the design process; different people have different expectations on the guidelines, also depending on the current task:

- The “prototyper” expects high-level advice, concerning overall application design, task guidance and dialogue layout, organized with respect to tasks and dialogue objects.
- The “detail designer” expects design rules including definitions of interaction object selection strategies, consistency demands, style guide excerpts and local guidelines, organized by user tasks and application functions.
- The “implementor” expects low-level guidelines, referring to widgets, key-codes and terminology, organized by interaction objects and interface architecture components.
- The “evaluator” expects the information to be organized for evaluation of the particular application, user group, user environment and tasks.

Guidelines that fail to take into account some crucial role or task in a project are likely to be perceived as too narrow and less useful. If, on the other hand, a document is prepared to deliver information for several dif-

ferent stakeholders and tasks, the content is more likely to be contradictory (Hammond et al., 1987), hence less useful.

A corresponding issue is the *specificity* dimension; how general or specific is the guideline content? If the material is made too specific, it will be hard to adapt to new situations. A very general material tends to be equally hard to adapt. Several studies have shown that common guidelines are sometimes too general, and sometimes too specific (Chapanis, 1990, Hammond et al., 1987), to be easily and effectively used by the designer. Jeffries et al. (1991) show that when used for evaluation, guidelines helps identifying recurring and general problems. However, in their study, software developers using guidelines miss some severe problems.

Because of the problems the designers experience when using guidelines, they might perceive them as less useful, less valuable and less important in the project. Even worse, they might consider the kind of *knowledge* represented in them as less important. de Souza and Bevan (1990) report that designers do not respect about 11% of guidelines.

Despite all those potential or actual problems, written guidelines are used in many projects, either because they are formally required or because they are considered useful. Consistency and style conformance are important issues for many developers (Nielsen, 1989), and the ability to transfer design experience to future systems development is valuable. The emerging requirements for standard conformance emphasizes the importance of guideline information. Hence, there is a need for improving the guideline approach; to support the use of guidelines in user interface design work.

2.4 Supporting the use of guidelines

Since design knowledge in the form of guidelines are often used (or at least intended to be used), and judged important or useful, there have been several attempts to overcome their difficulties and potential problems. Different techniques has been tested to overcome the information-handling problem. In essence, the approaches tested in previous research resemble five different technical solutions, described below.

Note that we use the technical approaches on a conceptual level only—for the purpose of describing classes of tools—the specific techniques may have been intended to address other problems originally.

2.4.1 Implicit compliance constraints and affordances

Implicit compliance control can be achieved by integrating well-defined constraints in the tools used; the support system is equipped in a way that affords certain design directions and constrains the designer in some sense. This can be implemented by adapting the widget set, the mechanisms for combining widgets and/or the layout functions in a way that implicitly forces the designer to conform to a certain set of rules (style guides, guidelines, standards, etc.). The common “snap-to-grid” mechanism found in most layout tools is an example of a very simple implicit compliance constraint.



Figure 2.5 Design support via (implicit) constraints: providing predefined widgets and compound interaction objects in the TeleUSE UIMS (Alsys Inc., 1994).

The constraints need not be parts of the system only; they may be generated and controlled by the designer via templates and reusable components. In several systems it is possible to build databases and directories of design components; compound widgets, complete dialogues and even adaptable interface templates (see figure 2.5).

Using constraints and affordances in the tools has the advantage that consistency and style conformance are rather easy to achieve. The potential drawbacks of this approach are the risk that the designer perceive the constraints negatively, that the constraints inhibit creativity and that only low-level guidelines can be handled this way.

2.4.2 Hypertext systems

Another passive approach is the use of hypertext systems: the rule set (style guides, guidelines, standards, etc.) is stored in a hypertext system, making it possible to browse and possibly search and organize guidelines. These systems are often generated from paper originals, by cross-referencing the material and inserting hypertext links for keywords. The main interface often starts with a hypertext linked table of contents, presenting an overview of the guidelines from some arbitrary perspective (e.g., interaction objects, dialogue types or user task types).

NaviText SAM by Perlman (1988) contains a subset of the Smith & Mosier guidelines (1986) and allows browsing, searching and annotation. Cohen (1995) used a subset of the MIL-STD-1472 standard (DoD, 1974) to create the contents of the hypercard stack MIL-STK-1472. BRUITA-SAM, HyperSAM (Iannella, 1994, Iannella, 1995, see figure 2.6) and GUIDEBOOK (Ogawa and Ueno, 1995) all contain subsets of the Smith & Mosier guidelines (1986). Fischer et al. (1989) included a hypertext component in the knowledge-based environment JANUS. This provided the designer with the opportunity to browse preventively, or to follow up feedback from the system (see commenting systems below). Vanderdonck (1995) identifies the need to reorganize the guideline documents and provide new structures for presenting them. In SIERRA, he demonstrates a representation model that allows the designer to browse, search and prioritize the guideline set using different ergonomic criteria and levels.



Figure 2.6 Hypertext design support: the HyperSAM tool by Iannella (1995).

Compared to the written original, this solution offers more flexibility for managing and accessing the information in the documents (Reaux and Willeges, 1988). In a study by Fox (1992), it is also shown that at least some groups of users prefer using on-line guidelines over the paper-based format. However, in the same study, no significant differences in productivity were found. The approach requires the designer to actively search for and adapt information, which may afford learning, but not necessarily productivity.

The major problem with most hypertext tool approaches is that they are not really integrated in the design environment (Bias et al., 1993). Other potential drawbacks of this solution are the problems constructing the information space and the navigational system. Hypertext systems always bring the risk of users getting lost while looking for some piece of information, if the space is large and the structure complex (Fischer and Nakakoji, 1992). Using hypertext systems doesn't automatically solve the problems with information overflow or contextualization.

2.4.3 Example databases

Instead of forcing the designer, an informing or educational approach may be taken by providing examples of good and bad design during the design work; the style guide and "design rules" are made implicit in the examples. These types of systems are mostly passive, i.e. they require the designer to actively browse and search for suitable examples. Example systems are in effect a type of *reuse* systems, i.e., they make previous design solutions and experience available for the designer to use (and adapt) in new design situations. The effect may be either physical or conceptual reuse. In the first case, the designer merely copies and adapts previous design components. In the conceptual case, the designer uses design ideas and experience to create a new solution. Of course, the alternatives are intertwined.

In KID, by Fischer and Nakakoji (1994), the system provides a catalogue of previous designs in which the designer might retrieve information using similarity metrics, to gradually narrow a catalogue space during design. The catalogue is adapted (ordered) by the system to present cases relevant to the current situation (Fischer et al., 1995). The IDA Advice Tool, by Reiterer (1994, see figure 2.7), has two different kinds of examples: widget/dialogue component usage definition and pictorial guideline examples.

Of course, this kind of system can be combined with hypertext systems. The IDA Advice tool guideline system is a hypertext system that contains pictorial examples. SIERRA, by Vanderdonckt (1995), is a hypertext system enhanced with pictorial examples of positive cases and exceptions. The latter approach, to store exceptions or negative examples may be crucial to encourage the designer to learn what went wrong in the past (Fischer and Nakakoji, 1992).



Figure 2.7 Design support via example databases: the IDA Advice tool by Reiterer (1994).

The example databases add to the features of the hypertext the advantages of using pictures for delivering information. Of course, this also brings the potential problems that come from the extra effort required to produce the material, and the risk that the designer uses only the pictures and disregards the textual information, as shown by Tetzlaff and Schwartz (1991). However, they also report that designers find examples helpful during user interface design. Blatt and Zacherl (1993) presents suggestions and requirements from a prototyping session, aimed at developing an example system.

An attempt that in some way addresses the problems of the early example and hypertext systems is presented by Henninger et al. (1995). In *Mimir*, a system for supporting the development and evolution of guidelines in a design organization, the designer can associate guidelines with design cases to make them contextualized and easier to find. The knowledge stored and maintained can be displayed and accessed in a hypertext-like form.

2.4.4 Automatic Design Systems

Instead of just providing information or constraining the designer, the tool can also be made *active* and *generative*. In an automatic design system, some kind of formal specification of the target system is used as input, together with a specification of the desired look and feel. The tool then automatically creates one or more designs: the rule set drives the widget selection and layout mechanisms in the system. The designer may use such a tool to create a set of different basic proposals of a layout, and then select some of them and perform manual refinement (improver-based design).

In Mike (Olsen and Halversen, 1988) a semantic specification of the commands that the interaction is to support is used to generate the syntax level of the interface. DON (Kim and Foley, 1993) uses lexical and syntactic rules to automate design of dialogue boxes, based on a model of the application's data and control structures. The system generates several alternatives, from which the designer can select the "most preferred" solution. The ITS system (Wiecha and Boies, 1990), which was used to create the visitor information system for the EXPO 1992 worlds fair in Sevilla, lets the designer specify the dialogues and their content separately and apply style rules generating the layout later on. The Mecano system (Puerta et al., 1994) generates not only static layout from data models, but also dynamic behaviour from a high-level domain model. Other examples of automated systems are: ACE by Johnson et al. (1993), UIDE (User Interface Design Assistant) by Sukaviriya et al. (1993), HUMANOID by Szekely et al. (1993), MASTERMIND by Neches et al. (1993), ADEPT by Wilson et al. (1993).

The effects of automated design systems are similar to those of the affordance/constraint techniques: effective management of consistency and low-level design, but an increased risk for negative reactions from the designer (due to perceived loss of control). The benefits of rule-based systems where look-and-feel can be treated separately and switched are obvi-

ous in projects that span several platforms and/or different customer requirements.

2.4.5 Evaluating/Commenting systems

Other active, or semi-active tools, are the systems that evaluate the designer's work and present feedback. The guideline set is embodied in an expert critiquing system, capable of analysing and evaluating the work in the design environment, either integrated or stand-alone. The guidelines are stored in some sort of rule set, which is used to evaluate the design, and to generate feedback. The designer receives the information and may choose to act upon them, or reject them. Examples of this kind of system, which are usually called critiquing systems or commenting systems, are given in section 2.6; figure 2.8 shows a snapshot from the PROTÉGÉ-II system.

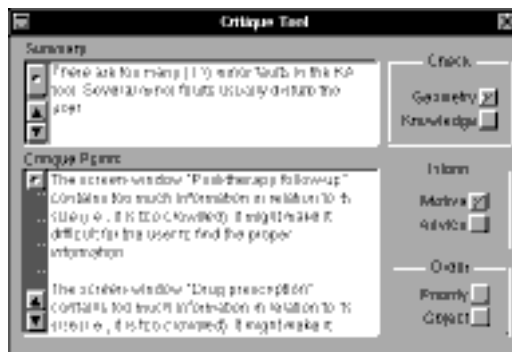


Figure 2.8 Design support via commenting systems: the UI critiquing component of the PROTÉGÉ-II system (Eriksson et al., 1996).

In comparison to the other approaches, commenting systems add the possibility to take into account the context; to make only the relevant information available and to adapt it to the situation. The designer gets help to filter out relevant guidelines and to interpret them for the current task, but stays in control of the design action. Some systems enhance the evaluative mechanism with a generative component, which can automatically remedy the flaws.

2.5 Commenting Tools as DSS

A commenting system is a knowledge-based system that evaluates a user-generated solution to a problem, with respect to certain criteria, and delivers comments to the user. This type of system is known as “critiquing” in the AI community, where it is typically described as a rule-based system of some sort. The area of critiquing and commenting is rather large, and spans several application domains. We do not describe the whole area but focus on commenting tools in the user interface design support area. For a more general description of critiquing, we refer to the overview by Silverman (1992a) and the special issue of Knowledge Engineering Review on critiquing (1993). Hägglund (1993) describes critiquing as follows:

”It is assumed that the user initially proposes a decision or course of action. The system then reviews this suggestion relative to known circumstances, tries to evaluate the proposed solution, provides suggestions for improvements, draws attention to possible risks, indicates alternatives, and evaluates their merits, etc.”

In other words, it is the user who is in control; the commenting system presents comments, critique and alternatives, but it is the user who chooses to act upon the feedback. This non-generative characteristic of commenting tools is described on the “Design paradigm page” (MIT Media Lab, 1994):

“Critics are user-invoked agents that respond to user-generated graphical objects by providing criticisms. The range of criticisms offered can be broad, from notification of low-level geometric constraint violations to high-level critiques of object conceptualizations. The main characteristics of this paradigm are:

- the user is still required to perform the entire modelling task manually.
- critics identify flaws in design and articulation, but remedies must be applied by the human collaborator.”

Hägglund (1993) describes three different types of use of commenting systems:

- Used as a mechanism for reasoning and problem solving. The main use is as a way of organizing the knowledge and inferencing needed.
- Used as a way of providing non-intrusive advice to a computer user. The main focus is on the process of interaction between the system and the user.

- Used as the basis for presenting arguments and explanations in an understandable and effective way. The main focus is on the contents of a critique.

The knowledge in a critiquing system is often represented as state-driven condition-action rules, but other techniques exist.

Although we see the term “critiquing system” or “critic” above, we will continue to use “commenting” throughout this thesis.

2.5.1 Arguments for using commenting tools

The benefits of using the commenting approach in design environments have been presented and argued for in several previous reports (Fischer et al., 1991b; Silverman and Mazher, 1992; Fischer et al., 1993b; Löwgren and Laurén, 1993). We will just make a short overview of the advantages that are usually associated with commenting systems.

One of the primary advantages associated with commenting systems is that they are useful in situations where the problems in the current task are hard to specify, as is the task itself. Fischer (1993a) argues that commenting systems help when “knowledge about the design domain is incomplete and evolving”, “the problem requirements can be specified only partially” or when “necessary design knowledge is distributed among many design participants”. Silverman (1992b) focuses on performance features and argues that commenting systems are beneficial where users commit the type of errors that machines are good at eliminating. Specifically, he writes, critics can cause the user to notice and use more cues.

Commenting systems provide a viable approach for presenting relevant information to the user. By putting the design information in the computer, the designers get closer to the material; they do not have to 1) try to remember which book or folder that is relevant, 2) find the book or folder. However, the material is just available, not really integrated in the design environment (Bias et al., 1993). One of the basic principles of a commenting knowledge-based system is also that the user should be given relevant information; she should not have to search the complete source for the piece of information that is applicable in the current situation.

Looking into other domains, related claims have been made. In cases where the knowledge stored is incomplete or semi-structured, support systems that critiques the user’s work have been claimed to be more useful than semi-automated systems, or traditional expert systems. In an informal study by Guerlain and Smith (1993) users supported by a critic performed

better (fewer errors on a task) than those who used a traditional expert system, in the domain of antibody identification.

There are indications that people have a positive attitude towards commenting systems. Kluger et al. (1991) report that a computer is more likely to be consulted as a source for feedback than a person. This finding is supported by other studies, performed in the design domain, albeit small or informal (Löwgren and Laurén, 1993, Nakakoji and Fischer, 1995).

2.5.2 Effects of introducing a commenting tool in a user interface design environment

Previous research has presented a large number of potential advantages of using commenting tools as design support. Although most of the claims are supported by informal studies or observations only, we choose to present three of the most important issues here: retrieval of information relevant to the task at hand, learning on demand and reflection in action. These concepts are intertwined, which is evident from the descriptions below.

Retrieval of information relevant to task at hand

One of the primary ideas behind the commenting tools is that they help provide design information relevant to the task at hand (Fischer and Nakakoji, 1991; Fischer et al., 1990). In its simplest interpretation, this means that the feedback contains information that has relevance for the current situation. With a stricter perspective, it also implies that the feedback is not merely information of relevance, but knowledge of importance to and adapted to the task. This could mean that the knowledge represented in the system is not presented as raw information, but instead used, in combination with information about the work and artifact, to generate the content and form of the comments.

Closely related to this is the discussion about whether design support tools need to be domain-oriented or not (Fischer, 1994; Fischer and Nakakoji, 1992). Without domain knowledge in the system, it is hard to produce contextualized feedback.

Learning on demand

Another important claim is that commenting tools afford learning on demand (Fischer et al. 1991, 1992, 1993a; Harstad, 1993). This type of training or learning is claimed to be an effect of the designer's interaction

with the system; by actively using the knowledge, rather than passively perceiving it, learning occurs easier (Harstad, 1993). The claim relies on the assumption that by letting designers see for themselves the usefulness of the new knowledge, for a particular problem, the designers' motivation to learn new things will be increased (Fischer, 1991). Even if the primary concern of the commenting system is to improve performance or product characteristics, some learning possibly occurs as a by-product of the interaction between the designer and the system (Reiterer, 1994). These arguments refer primarily to cognitive theory and theories about situated cognition (Winograd and Flores, 1986; Suchman, 1987; Lave, 1988; Greeno, 1989), where it is stated that learning is highly dependent on the current situation. It has been shown that self-directed and intentional learning are important factors of effective learning (Brown and Palincsar, 1989; Bereiter and Scardamalia, 1989). Gillan and Bias (1992) argue that designers may learn human factors concepts by receiving feedback about their work, just as people using grammar checkers may infer grammar rules from using the tool.

In a small study of subjects using a commenting system, Nakakoji and Fischer (1995) report that making feedback relevant for the task at hand encouraged the users to explore the information in the system. They interpret this as an effect of the situation where the system delivers task-relevant, but not perfectly right, information and users feel that they know the answer; there is some evidence that people search longer for answers to questions when they believe that they know the answer (Reder and Ritter, 1992). This prolonged information browsing might lead the designer to new knowledge and insights, although it doesn't improve immediate performance and productivity.

A responsive prototyping medium

Effective design is achieved in an environment which allows "reflective conversation with the materials" (Schön, 1983); the design is a process in which the designer interacts with domain knowledge, via the artifact. Winograd (1995) calls a medium (design environment) that supports this activity "a responsive prototyping medium". However, artifacts themselves often do not "talk back" sufficiently to the designer, especially not in a user interface design environment where the "materials" are low-level interface objects and windows. Higher-level objects, originating from the domain and well-known to the users of the artifact, are needed. In Schön's description of artifact feedback, he refers to human agents becoming a

voice of the design situation. Computational critics have been claimed to remedy the same dilemma by becoming a voice of the design situation (Fischer et al. 1992, 1993a). By making the system provide domain-oriented feedback and allow specification in a domain-related language (visually or textually), a richer conversation is achieved. A claim related to the two previous ones is that commenting systems may support reflection in action (Fischer and Nakakoji, 1992), which is described as an important principle of design (Schön, 1983).

Product or process improvement

When it comes to the product of the design process, the user interface and the rest of the software system designed, several proposals concerning the benefits of commenting support systems have been made, but there have been no studies of the actual outcome. Most claims concern consistency and lower-lever interface aspects, e.g.:

- internal consistency, i.e., the design of the different parts of the application is consistent.
- external consistency, i.e., the design of the application is consistent with other applications from, e.g., the same company or on the same platform.
- guideline, standard or specification compliance, i.e., the system is in effect a sort of checklist that helps the designer perform verification.

2.5.3 Possible Drawbacks of KB Support Tools

We identify two, important, possible drawbacks of knowledge-based support tools in design situations: 1) dilemmas because of trust and 2) constrained creativity.

The main problem addressed by including a KB support tool is the overflow and complexity of information available to the designer. However, if a tool is used to filter out and present information, the designer has to decide whether or not to trust the tool. In many cases the information may be new to the designer, in others, it might contradict the designer's previous knowledge or intuition. This issue is discussed in more depth by, e.g., Wærn and Ramberg (1990) and Löwgren (1991), who also proposes the use of explanations (rationale) in order to overcome the problem. The question about trust also relates to the designer's freedom and creativity, which is the second potential problem area.

It is easy to argue that a KB support tool—consultative, commenting or generative—would inhibit the designer’s creativity. The designer might feel as if she is being forced to conform to the rules implicit in the tool. However, this need not be a problem inherent in the KB support tools; there is nothing forcing the users of such a tool to limit their creativity. The fact is that most user interface design environments create far more explicit limits, for several reasons. In some layout tools, the designer can only use a limited set of interaction objects, combine them using a predefined set of functions, and modify a limited number of visual attributes. Some tools are so complex that a designer only knows a limited set of the capabilities, and therefore adapts her design strategies to that. Furthermore, most tools (or project specifications!) do not allow the designer to create new, style-guide violating, interaction objects.

2.5.4 Characteristics of commenting systems in the design domain

A commenting system is characterized by its behaviour, which includes the approach to evaluate the target object, and the interaction with the user. Hence, defining and selecting specific behavioural aspects is important when designing a commenting system. It is also non-trivial, since there is a lack of guidance and results to refer to in this area; most previous research in the area of commenting systems for supporting user interface design has focused on feasibility and technical approaches.

Defining the behaviour dimensions of commenting tools is likewise difficult, but there have been some attempts to create a useful terminology. We present a list of some commenting tool behaviour factors below (the list served as input in the early studies of our research questions). This overview is based on descriptions and attempts to define the behaviour or collaboration factors of commenting and critiquing systems in previous reports by, e.g., Fischer et al. (1991b, 1993a), Silverman (1992a), Harstad (1993) and Stolze (1994). The different factors are intertwined in several cases, and are likely to lack generic “optimal” values. Instead, the designer of the commenting system must select and tune them for different domains, tasks, and types of users. Silverman (1992a) provides guidance and advice on this task.

Mode

A commenting system works in an *active* or a *passive* mode. An active system presents feedback autonomously as soon as it detects a rule violation. Using a passive tool, the designer has to ask explicitly for feedback. The mode factor is about control plasticity (Silverman, 1992a), i.e., where should the locus of the control reside?

In a study by Lemke it is shown that passive critiquing is often not activated early enough in the design process. An active system, on the other hand, has the disadvantage that it may disrupt the user's cognitive process and cause short-term memory loss (Fischer et al., 1993a). The perceived intrusiveness of the system is not only dependent on the mode, but also probably on the type of information delivered; non-informative, less useful information may be more likely to cause a negative impression. In the VDDE system (Harstad, 1993; Nakakoji et al., 1994) the user is allowed to tune the mode on a passive-active scale. Silverman (1992a) states that human-commenting system collaboration demands "plasticity", the ability to adapt the mode.

Timing

Commenting systems may use different timing to present the feedback. The presentation may occur *before*, *during* or *after* the problem situation arises. Stolze (1994) calls these alternative approaches early (preventive), immediate and delayed, respectively.

Timing is a most important issue, since it is the key to make the designer a) detect and remedy design flaws, and b) prevent future "errors". The problem is to present the right thing at the right time. If feedback is too preventive, the designer might not realize the importance of it (Miller, 1983; Langlotz and Shortcliffe, 1983), hence she might perceive it as disturbing. If, on the other hand, the feedback is always delayed, there is a risk that the real problems will never be solved but only the effects of them. A possible consequence of this is that the designer learns how to fix design flaws but not to design in order to prevent them.

Process

The process aspect defines the way the system handles the evaluation; whether it should be *incremental* or *batch-wise*. The incremental system runs whenever problems trigger them, i.e., they present feedback incrementally. A batch system processes problems and feedback in group, i.e., the feedback is presented batch-wise. Harstad (1993) uses a slightly differ-

ent perspective and the terms keystroke-dependent, conceptual unit and explicitly invoked to describe the process. Of course, this dimension is intertwined with the mode and timing aspect.

Design-time or runtime

A commenting tool may be a *design-time* system or a *runtime* system. In most of the descriptions above, commenting tools are described as an aid for the design situation. Therefore, it is often assumed that the input information has to come from the designer, guidelines, specifications, etc., which is also most common in previous prototypes. We call tools used this way *design-time* systems.

However, the designer's work is not finished just because the first prototype is released for user testing; if some kind of iterative design process is used, the designer will be redesigning the system based on data from, e.g., usability tests. Therefore, interaction logs and usability test data may also contribute to the evaluating mechanisms in a commenting system. We call tools that can utilize this kind of information *runtime systems* (Olsen and Halversen, 1988; Löwgren, 1991; Löwgren and Nordqvist, 1992).

Knowledge

The system behaviour is strongly dependent on the type of knowledge used for the evaluation control. First of all, the guidelines represented may be *positive* or *negative*, i.e., they may specify either what is allowed and encouraged, or what is disallowed. The knowledge may also be *shallow* or *deep*, i.e., refer to low-level maxims or high-level domain principles, respectively.

A shallow system has no knowledge about the origins or aims of the guidelines, the domain in which the designer works or the situations that occurs. Of course, this kind of system tends to become superficial. However, they are still considered useful in some situations, c.f., a spelling or grammar checker in a word processor. Including domain knowledge in a system makes it possible to create feedback that is more relevant for the task at hand. Higher-level feedback requires the system to "be aware" of parts of the designers task and the specification. For example,

- detecting terminology errors requires knowledge about the language and terminology used by the users.
- evaluating and presenting feedback about the organization of dialogue components and the dialogue structure requires knowledge about the users' tasks.

Fischer and Nakakoji (1992) argue that designers should perceive design as communication with an application domain, rather than as the manipulation of symbols on computer displays. For that reason, their system Janus (Fischer and McCall, 1989; Fischer and Nakakoji, 1994) incorporates several knowledge-based components with domain-orientation. However, low-level knowledge may be equally useful, as mentioned above. Shneiderman et al. (Mahajan and Shneiderman, 1995; Shneiderman et al., 1995) present consistency-checking systems, e.g., support tools that help validate spelling, terminology and physical layout aspects in a design.

Algorithm

The evaluation algorithm in a commenting system is either *analytical* or *differential*. In an analytical system, the evaluation is guided by a set of rules that in essence implements the guidelines that are to be used. If a certain condition arises, a rule triggers and presents a comment. The differential system, on the other hand, uses alternative design solution(s) to generate the feedback. The designers proposal is compared with the other(s), and the difference and variance is the source of the feedback. Most descriptions specify that the alternative(s) are generated by the system. Harstad (1993) uses the term consistency critiquing for a system where the alternatives are manually produced, to indicate the effect of the differential process in this case.

In both cases, the “implemented” guidelines may be positive as well as negative; the rules may specify what must not hold in a design, and the alternative solutions may exemplify sub-optimal or disallowed design directions.

Presentation form

One of the most important characteristics of a commenting system is the presentation form used to communicate the feedback to the user. This includes several aspects, e.g., the modality and cues used, the mood, specificity and argumentation, and the “interactivity” of the feedback. In other words, it is not just a question of “the right thing at the right time”, but also “in the right place and form” (c.f. Stolze, 1994).

Different *modalities* may be used for the presentation. The feedback may be pure text, text including pictures, voice or sound, graphical, etc. It may include cues with different degrees of connection to the object evaluated, e.g., sound signals, graphical annotations or flashing pointers.

If text is used for the feedback, this may be presented using different *moods*. The term mood is borrowed from linguistics, and describes whether the text is *imperative* or *declarative*. A declarative text merely points out the existence of a guideline violation. Imperative texts suggest how the violation might be remedied. Of course, this aspect is closely related to the knowledge dimensions, since feedback may also be *positive* or *negative*. In a positive system, feedback is encouraging, containing positive examples and preventive information. A system with negative feedback points out errors committed by the designer and shows negative examples. Silverman (1992a) includes these aspects in his overview of critiquing strategies where a positive, active system is called an influencer, and a negative, possibly imperative, system is called a debiaser.

The feedback may be more or less *general* or *specific*. Using the general approach, a comment may describe a design flaw in a dialogue in the following way: “the physical layout is too dense”. In the specific case, the same comment specifies what dialogue and what objects that are involved, and also the guidelines used to detect the flaw. This dimension is of course closely related to the knowledge and form mood aspects.

Another aspect intertwined with the knowledge and mood/specificity dimensions is *argumentation*; either the feedback is backed up with rationale, or the plain comments are presented. In studies by Fischer et al. (1991b) it is shown that users who do not understand the critics’ judgments will often follow the critic’s suggestion blindly. In those cases, argumentation is essential if knowledge is to be transferred to future systems development. In the Janus (Fischer and McCall, 1989; Fischer and Nakakoji, 1994) and IDA (Reiterer, 1994) systems, the system allows the designer to “follow” the rationale for a comment in a hypertext system.

Related to the argumentation aspect is something we will call “interactivity”. Most commenting research prototypes include some sort of interactive part, in addition to the basic feedback mechanism. The text may be enhanced with hypertext links to the sources of the guidelines or examples of those. Furthermore, the feedback may provide a means of using the examples, i.e., they are actually proposals of alternative design solutions for the objects causing the violation. All this means that the designer is not just a passive viewer/reader of feedback comments; she is encouraged to interact with the system in order to find more information about the guidelines.

Adaptive and adaptable systems

A factor that spans all the others is commenting system *adaptivity*, i.e., the system's ability to adapt its behaviour to the user (and possibly domain and task). A very simple kind of adaptivity, or system learning, is found in the spelling checkers of most word-processors today, where the user might "train" the system in different ways during a checking session: new words and spelling may be learned, and the user may choose to force the system to override or "forget" some words. In a user interface design commenting system, this could mean that the system was able to learn new guidelines and to override existing ones. This kind of mechanisms are found in systems by, e.g., Silverman (1992a) and Harstad (Harstad, 1993). More sophisticated adaptivity would imply the system's ability to adapt mode, process and timing.

A related dimension is that of *adaptability*, the user's ability to adapt the system manually. In most situations where a design-oriented commenting system is used, it is necessary to provide mechanisms for letting the designer modify the system's knowledge and behaviour. Modifications may include mode, timing or process tuning; rule definition and modification; presentation form modification, etc. There are several reasons for providing such mechanisms, related to the ill-defined, open-ended, iterative nature of the design knowledge and process, and the benefits of domain-orientation (Fischer et al., 1990, Fischer, 1994).

2.5.5 Summary of commenting system characteristics

The overview of characteristic dimensions presented above may serve as a tool for dissecting existing commenting systems in the user interface design domain, but more importantly, as an aid for developing and testing new systems. We do not claim that the overview presented here is complete; the taxonomies and descriptions presented by, e.g., Silverman (1992a) are more elaborate, and he covers a wider range of domains; Nakakoji et al. (1996) present ideas for the development of classification schemes for general support techniques in the domain of interactive system design. However, our dimensions are described from a perspective where user interface *design* is the central task, not problem solving, optimization or decision making. Table 2.1 summarizes the commenting systems characteristics described above.

	Dimension	Types
Process characteristics	Mode	active, passive
	Timing	early, immediate, delayed
	Process	incremental, batch-wise
		design-time, runtime
	Evaluation algorithm	analytical, differential, consistency
	Adaptivity & adaptability	adaptive adaptable
	Knowledge	positive, negative
deep, shallow		
Presentation (form) characteristics	Modality	text, image, sound, animation,...
	Mood	imperative, declarative
	Specificity	general, specific
	Argumentation	w/wo rationale
	“Interactivity”	

Table 2.1 Commenting systems characteristics dimensions

2.6 Evaluation and commenting support tools for user interface design—some illustrations

During the past ten years, several prototypes of evaluating and commenting support tools for user interface design have been built. The following overview is meant to illustrate the different approaches used in previous research and development.

One of the first evaluation systems for the design of software, and the first for user interface issues, was the Display Analysis system by Tullis (1984). Using Display Analysis, the designer could analyse alphanumeric displays and predict average search times and subjective preferences for their use. The system calculated a number of “goodness” values for the interface; those values had been tested and shown to correlate sufficiently with empirical data and subjective ratings. The output from the system consisted of a mix of the six characteristics and simple proposals of how to improve the interface.

The KRI/AG system, by Löwgren and Nordqvist (1990, 1991, 1992) is an analytical batch system for formative evaluation of graphical user interfaces. The system's knowledge base includes general principles as well as platform-specific rules, and is based on publicly available guidelines, e.g., Smith and Mosier (1986), Brown (1988) and the OSF/Motif style guide (1991). The designer creates an interface using the TeleUSE (Alslys Inc., 1994) UIMS, and feeds a description file into the KRI/AG tool, which then produces textual feedback.

CHIMES, by Jiang et al. (1992) is a batch system for evaluation of graphical user interfaces on the Motif platform. Like the KRI system it reads interface description files from an interface builder, in this case TAE (tae). The system is developed for compliance-checking, and the knowledge base consists of heuristics concerning graphical layout and colour usage. The layout information is based on the OSF/Motif style guide (1991).

The Framer system, by Lemke and Fischer (1990), is a system for supporting the design of window-based user interfaces. In this system, the authors focus on cooperative problem-solving and domain-oriented design environments, and the components required to achieve that. Design knowledge in Framer consists of rules concerning syntactic issues and consistency. The system can provide both mandatory and optional feedback, which the designer can explicitly reject, if she wishes. The system also offers a "remedy" mechanism which automatically provides a default solution for the problem at hand. For each feedback item, the user may request an elaboration on the rationale and solution. In the first version of the system, a passive mode was chosen, but, since observations showed that designer invoked the critic too late, a second version implemented a completely active mode.

Designer, by Weitzman (1992), is a tool that supports designers of interfaces for instructional systems. The aim of the system is to improve the quality of the interfaces by making them more consistent and visually more effective. The system's knowledge is domain-oriented, towards visualization, and includes general high-level principles as well as application-specific standards. Designers using the system may create, generate alternatives from the original and evaluate solutions. This reactive approach, Weitzman argues, makes the system support a more user directed explorative design process. Comment feedback from the system is presented in textual form.

The AIDE (semi-Automated Interface Design and Evaluator) system by Sears et al. (1993, 1995) allows designers to design, evaluate and redesign a user interface in one single environment. The tool includes two knowledge-based components, one for automated design and one for automated evaluation, both using layout appropriateness metrics (Streveler and Wasserman, 1987, Vanderdonckt and Gillo, 1994, Bodart et al., 1994).

In USAGE (UIDE System for semi-Automated GOMS Evaluation), Byrne et al. (1994) has enhanced the UIDE user interface development system (Sukaviriya and Foley, 1990; Sukaviriya et al., 1993) with an evaluation component. The evaluation results in execution and learning time estimates from a NGOMSL model generated by the system.

Diades-II, by Dilli et al. (1995), is an object-oriented knowledge-based system for prototyping of graphical user interfaces. It uses a blackboard approach which integrates several supporting agents, e.g., an assessment component that can evaluate the current design objects with respect to a set of guidelines.

In the graphical user interface editor prototype ActorStudio, Koivunen et al. (Koivunen et al., 1993) has integrated a critiquing component, Critic. A special-purpose language is used to store interface descriptions in ActorStudio, but may also be used to define design rules that guide interface evaluation.

SYNOP by Kolski et al. (1991) is a system for static ergonomic evaluation of graphic industrial displays. It is a batch system which contains ergonomic knowledge formalized into production rules, which can be used either to produce comments, or to automatically improve the current design.

Shneiderman et al. (Mahajan and Shneiderman, 1995; Shneiderman et al., 1995) are developing tools for checking low-level aspects of user interfaces, e.g., detection of anomalies in colour or fonts, variation in capitalization, abbreviations and terminology. They argue that such tools are useful, since "inconsistencies in spatial and textual style of an interface designed by several designers' may result in a chaotic layout. Each designer may have different interpretation of terminology and uses his/her own style of abbreviations and computer terms."

VDDE-critics, by Harstad et al. (Harstad, 1993; Nakakoji et al., 1994), is a system for supporting designers of voice-dialogue interfaces, e.g., voice mail systems. It allows multiple system behaviours, and combination of several, conflicting sets of rules (pluralistic critiquing). The designer is provided with several mechanisms for tuning the system's

behaviour: the mode can be tuned using a scale passive-active, rules may be given priority, rules may be violated by explicit reject/override, etc. VDDE-critics also provides multiple evaluation algorithms. It supports both an analytical process, based on the rule sets, and a consistency evaluation, in which two manually produced design solutions are compared, and the differences pointed out.

In the IDA user interface design environment, Reiterer has included several knowledge-based tools, in a manner similar to the approach taken in Janus (Fischer et al., 1989, 1989; Reiterer 1993, 1994, 1995). However, the domain used in IDA is graphical user interface design for the Motif/Unix or Windows/DOS platforms. IDA includes a reuse component, a commenting component and a hypertext advice component. The latter two systems are interconnected to allow the designer to inspect rationale from evaluation feedback, and explore related design knowledge.

As part of the knowledge-engineering environment PROTÉGÉ-II, Eriksson et al. provide a design critiquing component for the user interfaces built in the system (Eriksson et al., 1996). The feedback concern layout issues, as well as flaws in the internal design of the system built with PROTÉGÉ-II (e.g., inconsistent input ontologies). Comments on visual aspects are generated from a set of layout heuristics.

Later work from Fischer and colleagues focus on aspects like collaboration, multiple perspectives and conflict management, step-wise formalization of design knowledge, etc., and has resulted in several prototype systems. At the same time they investigate areas other than user interface design, hence we only consider some of the systems here.

The Hyper-Objects Substrate system is primarily a system for incremental formalization (Shipman III and McCall, 1994) and collaborative design. However, its object structure may contain agent objects, which can be used as critics. In eMMaC, Nakakoji et al. (1995) use experience from earlier research on knowledge delivery and elicitation in a support system for colour selection and combination in a multimedia development environment.

Chapter 3

Research Approach and Experiment Design

What is the appropriate way to test new techniques and tools for user interface design? There may be several different approaches depending on the research aims—the method and approach presented here matched the specific settings in our project.

Our primary interest was the usefulness and effects of using different commenting strategies. In order to deal with the question of usefulness, we believe that it is necessary to perform empirical studies with users working with a commenting tool in a design environment. In order to create useful knowledge that can guide the design of new tools, an ultimate goal must be to build and evaluate prototypes integrated in the work environment and actual tasks.

During earlier stages of research, prototypes may serve as “objects to think with” (Fischer and Nakakoji, 1994), i.e. artifacts that can be discussed and criticized in a concrete manner. A great deal of practical and explorative work has been reported (see chapter 2), and several tools with commenting capabilities have been built in other domains. However, we were not aware of any related earlier results (reported empirical studies) to build on, nor did we have a prototype commenting tool to work with (c.f. section 4.1).

We did not have the resources available for testing a new tool in a professional setting, and that kind of approach would have been neither feasible nor practical. In order for that kind of study to 1) result in usable research data and 2) not to interfere in a negative way at the participant site

we believe that the research questions, theories and variables studied must be well known and specified.

In the situation we were in, we believe that a feasible research approach is to:

1. Collect enough information about the issue of interest to make it possible to formulate a first set of research questions. In our case we believe that it is important to collect data in professional settings, in addition to previous research results.
2. Implement the questions in an artifact so that they can be answered, i.e. let the questions guide the design of a prototype artifact (of course this approach is not viable for all types of questions). At this stage, it is hard to raise funds and resources in order to run field experiments, therefore the experiments are performed in a lab setting. This has the drawback that the results are achieved in a non-realistic environment. However, there are advantages with this approach too. First, this lets researchers control the environment, thereby possibly improving experimental validity and reliability. Secondly, we believe that the research will benefit from using method triangulation, i.e. combination of different methods and data collection techniques; one approach can be used to facilitate the use of another and the two together may give a better general picture as well (Bryman, 1988). This step may provide the researcher with quantitative (performance and perception measurement) as well as qualitative (interviews and observation) data.
3. Use the results to formulate a theory for the design of the artifact; the answers and data from the previous step serve as design specifications and guidelines. Test the new artifact in an industrial setting, in a context as realistic (with regard to users, tasks, surroundings, etc.) as possible. This stage is used to collect data that can be combined with the primarily quantitative data from stage 2, and finally be used to better understand the real problems as well as answering the research questions.

At stage 2, it is also of utmost importance to be able to generate results that can be used to guide future research. Therefore, in the case of a negative result, a plain refutation of early hypotheses is not enough. In that situation we want guidance towards alternative solutions, without having to build a completely new theory and artifact that can be tested.

By using simulation techniques instead of real, complete implementations, it is possible to switch easily between several alternatives, and to

make comparative studies of different instantiations of a theory. We believe that the Wizard-of-Oz technique is a viable approach for simulating this kind of tools.

3.1 Objects to think with

Other researchers have provided results in terms of exploratory data, or technical implementations that prove the feasibility of a certain algorithm or solution. Fischer et al. describe their practical work in the area of critiquing support tools as building “objects-to-think-with” (Fischer and Nakakoji, 1994). By creating demonstration prototypes it becomes possible to test theories in practice, reformulate experiences in terms of new theories, and then refine the implementation accordingly.

Since a great deal of the prototypes built in the user interface design support area introduce completely new technologies or support approaches, it is often considered sufficient to perform informal user testing or even merely to demonstrate a tool. However, the object-to-think-with can also serve as a platform for evaluating a technique more formally. Even if the object is only a prototype, user testing is possible.

3.2 Design and evaluation of design support tools

The task of designing and evaluating design support tools is complex, and something that has not been reported on very often in HCI research. There are few usability studies, empirical investigations or methodological papers. User interface design and support tool development have a rather short history, and most of the previous work focuses on technical issues, feasibility studies and demonstrations. There is a lack of studies of users, i.e., the designers using the design support tools, their tasks and the context in which they work. The importance of, and need for, more thorough analyses of the use of design support tools is discussed by, and to some extent investigated by, e.g., Hix et al. (Hix, 1986; Hartson and Hix, 1989; Hix and Ryan, 1992), Morris (1987), Carey (1988), Bonnardel and Sumner (Bonnardel and Sumner, 1994; Sumner, 1995), and Redmiles (1995).

In a study of a small group of designers using the AIDE user interface management system, Hix (1986) investigates the usefulness of such tools, using task completion time (performance). The quantitative results, in combination with qualitative data, indicates that interactive user interface development tools can improve designers’ productivity. Carey (1988) presents a number of findings from informal observations of design tools.

Besides a focus on support for error message creation and help management, it is reported that job satisfaction is at least as important as efficiency and effectiveness. A related discussion is presented by Morris (1987), who claims that if an aid is perceived as being useful, then it will be used. In her overview of factors that influence acceptance, she includes the dimensions:

- perceived usefulness, interpreted as job performance.
- perceived usability, interpreted as “ease of use”.
- perceived level of discretion, interpreted as a mix of intrusiveness and user control.
- perceived organizational attitudes, interpreted as group attitudes.

She stresses the importance of encouraging designers to be more evaluation oriented, and suggests that the design tools should help achieve this goal.

Rosson et al. (1987, 1988) studied almost two dozens of design projects and present a view of the design process. They use the three phases information gathering, information processing and solution evaluation to categorize the process, and propose different approaches for supporting these. Their findings include conclusions about the importance of job satisfaction similar to those of Carey. They also discuss the need for different types of tool approaches: unconstrained, creative environments vs. structured tools that guide the designer towards an implementation. Another study of design projects and design practice is presented by Bellotti (1988). Besides describing a development model similar to that of Rosson et al. (1987, 1988), she reports on some of the problem sources in the process, e.g., uncertainty about requirements, unfamiliarity with the task domain and lack of guidelines and standards.

Hartson and Hix (1989) stress the importance of supporting all the activities carried out in the design process, and propose the holistic “star” life cycle for user interface development. In the life cycle model, they recognize the need for interleaving synthesis and analysis during design. Hix (1989) proposes a procedure for evaluating user interface development tools. The procedure, based on a questionnaire, leads to ratings of the tools’ functionality and usability as perceived by the users. Hix and Schulman (1991) discuss the method, and evaluate it in a small study, by applying it to a set of user interface prototyping tools. Results show a significant reliability for both rating measures. A second demonstration is presented

by Hix and Ryan (1992), who applied the method to a set of common user interface development tools.

Desmarais et al. (1994; see also Keller, 1994) performed a survey on user expectations for interface builders, by sending a large questionnaire to a number of software developers in Canada. The results indicate a need for improved performance, reliability, rapid prototyping capabilities and documentation, but also a demand for tools that support error detection, interface standard enforcement and portability aspects. They also report on what developers perceived as the main obstacles when introducing interface builders in the development, e.g., lack of knowledge about such tools, lack of standardization and complexity of usage.

Myers (1994) raises some questions about the complexity of user interface development. The overview of potential problem sources is primarily a discussion, but includes empirical backup for some central claims. He touches upon the importance of support for testing/evaluation and need for improved guidelines.

In the domain of commenting or critiquing support systems for user interface design, there are few empirical studies and results; most claims result from rather small or informal studies. In a study of participants using the FRAMER environment (Fischer et al., 1991a; Lemke and Fischer, 1990; see also section 2.5.4) in a strictly passive mode, results indicate that users often invoked the system too late, i.e., the designer could create several critical design flaws. This, in turn, led to more errors before feedback was delivered.

Löwgren and Laurén (1993) conducted a small study of professional designers in order to investigate the need for and acceptability of commenting techniques as design support. The designers were given the task to design an interface using the TeleUSE UIMS. After the session, the designers were interviewed in a semi-structured manner, and their design evaluated using the KRI/AG commenting system (Löwgren and Nordqvist, 1992, see also section 2.6). The results indicate a need for better ways of handling guideline knowledge, and also that designers would find a commenting tool valuable. As for strategies and system behaviour, the designers want the system to leave them in control, and to indicate the severity of the detected design flaws.

Bonnardel and Sumner (1994) report an exploratory study of two professional designers using the VDDE design environment for phone-based interfaces (see also Harstad, 1993). The designers were given the task to design a new service into an existing system, using the design environ-

ment. A combination of active and passive commenting strategies were used. After the session an open-ended, structured interview was conducted. The results reported indicate that the commenting component had little influence on the resulting design, but also that the designers felt stimulated by knowing that such a component was active. This knowledge made them analyse in depth why they were violating guidelines, and, as a consequence, to record their design rationale in the system's knowledge-base.

In other domains, e.g., mechanical or engineering design, software engineering and electric circuit design, there exist several studies, formal and informal, of commenting support tools. Malinowski and Nakakoji (1995) report briefly on a study of users of the KID design environment. The system, which is not for user interface design but for kitchen floor-plan design, contains several knowledge-based components for delivering information to the designer. A catalogue of previous designs provides examples, a critiquing component delivers comments on the current design, and a knowledge structure that represents design rationale presents design argumentation. Observations of ten participants produced results similar to those of Bonnardel and Sumner (1994). Comments from the system were found to encourage the participants to reflect on their partial design, and also to react and argue against delivered knowledge in terms of the current situation. A related finding was that participants were encouraged by the knowledge delivery to explore the rest of the information space in the system. This behaviour is explained by the authors as a consequence of the indication that people search longer for answers to questions when they believe they know the answer (Reder and Ritter, 1992).

A larger set of empirically collected results from domains other than that of user interface design is presented by Silverman (1992a, 1992b, 1992c). However, the studies behind those results have had other issues in focus, such as decision-making, problem-solving and performance.

To summarize this section, we used the results and conclusions above as guidance when refining the research questions and designing an experiment to investigate them. Among the factors of most importance to us were:

- the use of empirical studies of the situation in which support tools are to be used.
- the importance of measures of perceived usefulness and job satisfaction, in addition to performance and productivity data.

3.3 Background study

Prior to the CODEK experiments, an exploratory study of professional designers was performed to gain an initial understanding of the phenomena of interest. The aim of the pre-study was twofold: to guide our selection of research questions, and to help us identify measurement points in this study.

3.3.1 Participants

A total of eight professional designers from local software companies participated in the background study. All of them were experienced user-interface designers, having spent between 2 and 5 years in the area. They were invited (in pairs) to participate in the study, and were paid by their companies for the time spent on the experiment.

3.3.2 Procedure

Each pair of designers was asked to collaborate to solve a design task. They were instructed to work for at most one hour, after which an experimenter would interrupt them. The task was presented in a short, written specification which described a telephone database application for internal company use.

During the task, the designers used paper and pencil to design the system. They were allowed to talk and discuss the task freely. An experimenter sat nearby the designers, observing them, and answered questions about the design task. Questions about the work process or design approach were not answered. The whole process was recorded on videotape, and the tapes were transcribed later on.

3.3.3 Data analysis and results

From the videos and tape transcriptions we identified commenting behaviour in the communication between the designers. In all the groups, designers took on the role of being “commenter” every now and then. Sometimes this role lasted for a couple of minutes, sometimes as long as 10–15 minutes. Initially, the most experienced of the two designers acted as the commenter, while the other designer sketched and generated ideas. The designers then switched roles back and forth, one being the commenter, the other the recipient of comments. This behaviour was found in *all* the participant groups.

We found that the commenting designer behaved primarily actively, delivering comments without being asked to do so. The comments were often illustrated with examples, and mostly on the imperative form, i.e. the designer told his colleague “how to do it instead”. Very seldom did the commenter back up his comments with references to documents or sources, e.g. “the guideline says that...” or “but the specification requires...”, although many statements obviously originated from such material.

Our observations are not at all revolutionary or unexpected. Nevertheless, we believe that in order to test the usefulness of commenting tools, one should be aware of the human approach to commenting in design work.

3.3.4 Our use of the exploratory study

The data and results from the exploratory study were used to inform the empirical study and specifically to guide our selection of research questions. Of course, the pre-study concerned human-human interaction, and from this we cannot draw conclusions about the situation where a human designer reacts to support from a computer-based design tool (Jönsson and Dahlbäck, 1988). Nevertheless, we believe that there are certain aspects of the design work that may be present in both contexts. Furthermore, before the pre-study, we had a very large set of interesting, yet vague, possible research questions concerning commenting strategies. By studying the real situation we created a (limited) understanding of it and at the same time gathered information that made it possible to prioritize and select among the questions.

From the tentative results, three findings served as guidelines. First of all, it is clear that a kind of commenting behaviour was used by all the groups. This is a motivation for carrying out the study itself, to see if the behaviour to discuss and refine solutions via comments would be useful if the dialogue partner is a computerized agent.

Secondly, the human commenter behaved in an active manner. The question of which kind of delivery approach should be used (active or passive) in the domain of computerized design support tools domain has been addressed in several practical studies, but not empirically analysed. What effects do the different approaches have on the designer?

Thirdly, very seldom did the commenter back up his comments with references to sources, or use direct quotes from guideline documents. This is also something that has been tested informally in different prototypes,

but not evaluated in comparative studies. Hence, the form of the comment itself is a crucial question—what is the best way to present the comments?

3.4 Using the Wizard-of-Oz approach

In research concerning large and complex systems there are several situations when implementing a complete system is not possible. Two of the most common arguments for a simulated system are:

- the system might be too costly to build. The research is performed in order to evaluate the usefulness and need for the system, before system design is started.
- the techniques required to implement the system might not yet be available, but within reach. The simulated study is performed in order to evaluate the need for further research.

We decided to use Wizard-of-Oz techniques to collect data, using a simulated environment instead of implementing a complete system. In this approach, the participants are led to believe that they use the real system, when in fact it is simulated by a human operator—the Wizard (Fraser and Gilbert, 1991; Dahlbäck et al., 1993). A typical Wizard-of-Oz experimental setup is depicted in figure 3.1. We had two major reasons for choosing this technique.

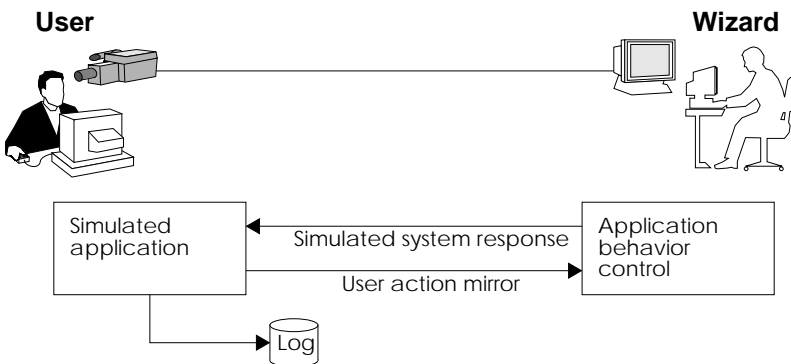


Figure 3.1 A Wizard-of-Oz experiment setup

First, as we argue above, the nature of our research problem is not suitable for full-scale implementation and evaluation. As we still want to collect more information about the situation, we need a tool that helps us observe different users, tasks, needs and behaviour as much as it supports

empirical data collection concerning our specific questions. Wizard-of-Oz environments make it possible to illustrate not-yet available/implemented systems (Coutaz et al., 1993a). In the HCI domain, it has been described as an experimental evaluation technique (Coutaz, 1994), or a rapid prototyping method for systems costly to build or requiring new technology (Landauer, 1987; Wilson and Rosenberg, 1988; Maulsby et al., 1993).

Secondly, this method has been shown to be a viable approach for similar research questions in other domains, primarily concerning the study of natural language communication with computers. The major areas of interest have been information retrieval systems and intelligent systems, e.g., expert systems, help systems, tutoring systems (Jönsson and Dahlbäck, 1988; Guindon, 1988; Carroll and Aaronson, 1988; Francony et al., 1992; Dahlbäck, 1992; Maulsby et al., 1993). Hill (1993) reports on a study, with a rationale similar to ours, where the purpose was to guide the design of advice-offering software.

There are few reports where the Wizard-of-Oz technique has been used in the domain of graphical user interfaces and interactive systems. Hauptmann (1989) let his participants manipulate graphical objects on the screen using spoken commands. NEIMO (Salber and Coutaz, 1993; Coutaz et al., 1993b; Coutaz et al., 1993a; Balbo et al., 1993) is a system aimed at studying multimodal interfaces with the capability of letting several Wizards cooperate to simulate the behaviour of a more complex system.

In order to analyse the effects of different strategies and the overall usefulness of the commenting design support system, we needed a system that:

- allowed potential designers to work on a realistic task.
- allowed us to switch between different system behaviour easily.
- was possible to build in the context of limited time and money.

The solution to use a simulated system fulfils these demands.

3.4.1 Ethical concerns

Of course, the technique has its disadvantages and potential problems. Is it really necessary to let the participants believe that they are interacting with a real system; must they be deceived? It might seem that role-playing would be an alternative to the Wizard-of-Oz approach. However, in the domain of natural language, it has been shown that people knowing about the Wizard behave differently and use different dialogue strategies than those who believe they interact with a real system (Dahlbäck et al., 1993,

Fraser and Gilbert, 1991). Therefore, it might be argued that the best strategy is to let the participants believe that they use a real system, but not by explicitly deceiving them. What the participant is told is one thing; what she believes is quite another (Fraser and Gilbert, 1991). This issue is discussed further in section 8.2.4.

3.5 Clarifying the research questions

This study commenced with the three main questions concerning usefulness, commenting strategies and knowledge transfer declared in the introduction. Although rather simple, the questions can not be addressed without a thorough specification and adjustment to the context.

First, to be able to discuss the usefulness of a tool, the situation in which it is used must be known. Defining the situation in this kind of study implies introducing boundaries and limits in a sense, since we need to be able to control certain aspects of it; the design task, surroundings and tools are therefore specified by us.

Secondly, in order for us to be able to measure knowledge transfer, the nature of the knowledge supplied by the tool and used in the work situation must be well-known. For this reason we use a limited design task, and a limited set of material for the comments to be delivered, i.e. the task and the guideline information are specified by us.

The following sections account for the designer's work context, as studied in our experiment, and the operationalization of the research questions.

3.5.1 Defining the designer's work situation

As indicated in the previous chapter (e.g., section 2.2), the user interface design process is complex. Since we are mainly interested in the support tools, this limited study captures only a certain phase, i.e., the practical design and implementation work using a graphical layout tool. It is assumed that this phase has been preceded by, e.g., task analysis, conceptual design and perhaps lo-fi prototyping. The designer has access to a written specification of the requirements, or at least the parts related to the user interface.

In addition to the specification and other material describing the task, it is assumed that the designer has access to guidelines and style guides. In order to make the commenting procedure as realistic as possible, and at the same time adapting it for our purposes, we decided to use a small, prede-

finer set of guidelines, but also let this set include several different types of information. In a larger project, information supposed to guide the design work consists of lexical as well as semantic guidelines, and general advice as well as project/domain specific information. Our approach to this was to integrate well-known (publicly available) guidelines and task specific information.

Furthermore, since we focus on the relation between a designer and his/her support tools, we do not specifically consider the situations where several designers share design work and cooperate actively via their tools. Our study is limited to a single designer working alone to solve a limited task. (This makes our work applicable for most teamwork too, since not all of that consists of object sharing, but working on different parts of a system.)

To summarize the limitations put on the designer's work situation in this study:

We consider a single designer, working alone, using a graphical user interface design tool to implement a graphical user interface. The designer's task is specified in a written overview of the requirements. A set of user interface guidelines, more or less adapted for the task, is available in some form.

3.5.2 Is a commenting tool useful in user interface design work?

The primary question addressed concerns the general usefulness of a commenting tool in user interface design work. As stated in the introduction, we use a limited set of concepts to define this; we evaluate the system from the point of usefulness as perceived by the designer. In addition we explore transfer of design knowledge, and the number of (and severity of) design flaws eliminated by using the tool.

We use the perceived usefulness of the tool as one of the basic measures of usefulness: If an aid is perceived as being useful, then it will be used (Morris, 1987). Unlike Morris, who specifically refers to job performance, we are interested in the designer's general perception of the system's usefulness, i.e., this is a purely subjective measure.

Other interpretations of the term useful is of course possible. One obvious way to define the value of this kind of tool is to include the quality of the resulting interface. However, that would focus more on the product than on the process, and also demand a very carefully designed method for

evaluating the resulting interfaces. Several factors, not easily controlled, affect the resulting interfaces, e.g., the UIB and the designer's competence. We chose to use a more basic quality measure by studying the elimination of design flaws and the effects on the designer's own commenting behaviour. Both factors are important for the outcome of the designer's work.

An important aspect of design flaw elimination is the conceptual level of the flaws eliminated. If they are mainly lexical and syntactic, corresponding to the kind of design knowledge found in style guides and guidelines, the support tool can be said to have great potential benefits considering the increasing industrial importance of style guide compliance. However, the elimination process must not induce more higher-level flaws, concerning semantics and tasks.

A possible consequence of experiencing someone helping you to improve your work, e.g., to eliminate possible flaws in a design, is that you adapt your own behaviour consequently; you learn from the aide. If this is the case, the *quality* of the transferred knowledge is of essence. Also, the understanding of the knowledge is interesting, since this allows the person to generalize to new situations. Hence, we wanted to evaluate the learning effects, or knowledge transfer, due to the use of a commenting tool.

In order to investigate the usefulness of a commenting support tool, i.e., to find a measure for its contributions in a design environment, we needed two sets of participants. One group of participants should use a design environment with an integrated commenting system, while the other group should use only the design part of the system. By comparing the first group with the second one, we would get a source for measures of usefulness. A small set of questions focused on the user interface builder only would help us identify and compensate for its impact.

The following specifications were added to the experiment design:

To measure the usefulness of a commenting system, two groups of participants are used: participants in group S (simulated) use a complete design environment, with an integrated commenting system, while participants group C (control group) use the design environment without the commenting component.

A questionnaire is used to collect information about the perceived usefulness of the system.

3.5.3 What is the most appropriate way to deliver comments?

A large number of different approaches to delivering comments, or critique, have been demonstrated in previous prototype tools (see section 2.6). Two of the most important factors of the delivery approach are the comment form and commenting behaviour (the look and feel of the commenting tool). However, those two factors are composed of a multitude of aspects, as indicated in section 2.5.4, e.g., timing, mode and knowledge type.

Based on the observations from the pre-study, together with claims and findings from prototype studies, we chose to study the two factors using the dimensions textual mood and delivery mode.

What is the most appropriate mood for comments?

The textual form of a comment is one of the most important aspects of a commenting system. However, to our best knowledge, no previous research in the design domain has examined the effects of different presentation models. Several early research prototypes implement rather simple presentation, displaying raw instances of the rules representing the guidelines. In later systems, it is claimed that rationale and examples are important, hence the (still rather telegraphic) comment is linked to a hypertext system with extensive examples and the full contents of associated guideline extracts.

After having explored the data from the pre-study, in combination with studies of reports of previous research prototypes (Fischer et al. 1991a, 1991b; Reiterer, 1994; Harstad, 1993; Stolze, 1994; Eriksson et al., 1996), we decided that we wanted to study differences between a) presentation of *information about* a flaw, and b) presentation of *the way to remedy* a flaw. We labelled the different approaches *declarative* and *imperative*, respectively, with terms borrowed from linguistics.

In order to study such differences, the two sets of texts must be comparable, i.e. a pair of texts must have the same content. We realized that it was very difficult to create different versions of each text without developing a huge set of canned texts; one declarative comment may have several different corresponding imperative ones (due to different objects, various attributes, etc.). However, we wanted to use canned texts only, in order to keep the performance of the Wizard rather high. An alternative approach would have been to create guidelines of our own; texts for which we could

create different presentations with the same content. Since we wanted to use well-known, publicly available guidelines, this was not an option.

The approach taken in this study was to test for differences between the mood of texts only, i.e. we use the terms declarative and imperative in their pure linguistic sense. We believe that if we can find differences on this level, a more sophisticated analysis of higher level differences may be justified.

The following specification was added to the experiment design:

Half of the participants in group S use a system where the feedback is purely imperative, while the other half uses a purely declarative system.

What is the most appropriate mode for a commenting tool?

Besides the form of the comment, the mode behaviour of the system is a very important factor. Active and passive strategies, and even combinations, or adaptable systems, have been demonstrated in previous research prototypes (Löwgren and Nordqvist, 1992; Lemke and Fischer, 1990; Fischer et al., 1991b; Harstad, 1993). Some studies have, informally, compared the effects of using different modes (Fischer et al., 1991a; Bonnardel and Sumner, 1994), but we are aware of no empirical results *explaining* the advantages and disadvantages in our domain. Our aim was to explore effects on user attitude, knowledge transfer and process impact.

In the case where a passive system is used, the designer has full control over the commenting tool. However, the potential usefulness might suffer, since there is a risk that the tool is used very infrequently, causing the designer to receive comments too late. All this might result in late fixes and last-minute modifications. If the tool is active, the control is moved from the designer to the system, causing the timing strategy to be critical. In this case, the tool might deliver comments in the wrong situation, i.e., the designer is not able to receive information. The result is a less usable system, but for other reasons than above. The designer might get irritated by the interruptions and therefore perform worse in the design work.

In order to compare different modes and evaluate the effects of them, we wanted to test the end-points of the scale: a completely active system and a completely passive system and collect data about the designers' attitudes and the flaw eliminating effects.

The following specification was added to the experiment design:

Half of the participants in group S use a system where the mode is purely active, while the other half uses a purely passive system.

Defining appropriateness

When addressing the form and behaviour questions, we aimed at investigating the appropriateness of different settings for each of the dimensions. For this measure, we reasoned that we would need several different sources: the designer's perception of the situation, the effects on the process and the results on the designer's future behaviour. Since we are interested in the perceived appropriateness, we decided to use a questionnaire to collect the data.

We also decided to test the direct impact on the designer during the process by measuring her perceived mental workload (MWL). It would be reasonable to assume that a higher perceived MWL, if related to the support tool, could make the designer rate the tool less positively. Hence, it would be interesting to measure the perceived workload in situations where comments had been given and compare it with 1) measures from other non-commenting situations, and 2) other commenting situations. This way it would be possible to investigate the perceived MWL as compared to an "ordinary" design environment. The data would also help us explore different combinations of types of design situations and perceived workload, which could possibly aid in developing more effective delivery strategies, i.e., find criteria for optimal delivery timing.

Physiological measures, e.g., using EEG, EKG, eye functions, and other stress measures were considered too intrusive and too technically complex. From a practical point of view—we did not have access to the technical apparatus needed for such experiments. Furthermore, the validity of such measures is still debated.

We decided to use a subjective workload measurement method, since we did not want any further interference or distractions in the participant's environment; intrusiveness was an issue in the study. Further arguments for selecting a subjective method were that (c.f. Eberts, 1994; Muckler and Seven, 1992):

- we were interested in point-wise measures.
- we were primarily interested in how the participants perceive the commenting system, not an absolute value of their MWL, or the capability to identify the attentional resource pool that is affected.

- such methods are more likely to result in participant acceptance, since they see it as a valid measure directly reflecting their subjective feelings of workload.

Since we wanted a rather simple method, we used a slightly modified version of the NASA TLX method, called RTLX (Hart and Staveland, 1988). The RTLX is easier to use, both for the participants and for the experimenters, and is essentially equivalent to the TLX scale (Byers et al., 1989). Since we did not want to interfere with the participants during the design task, we decided to measure the perceived MWL immediately after the design session, by replaying video sequences of situations we wanted to investigate.

The following specification was added to the experiment design:

Immediately after the design session, each participant is given a questionnaire in which system usefulness and presentation appropriateness are measured. After this, a mental workload measurement session is conducted, using prompted recall: the designer is shown video sequences of design situations of interest, and then asked to complete an RTLX form.

3.5.4 Does the use of a commenting tool imply knowledge transfer?

We wanted to examine potential educational value or process contributions from using a commenting tool. There are two reasons for exploring this issue. First, we consider the degree of usefulness of a commenting tool to be dependent on the learning effects. Therefore, studying the change in behaviour after using the tool was important. Secondly, we wanted to investigate the degree of knowledge transfer as such, since commenting tools have been said to support implicit learning or learning on demand (Fischer, 1991; Harstad, 1993). If this is true, such tools might be an interesting alternative (in terms of time and money) to other techniques for training designers, e.g., courses or written material.

Testing knowledge transfer includes more than just the *quantity* of knowledge, the *quality* is equally important, as is the ability to apply the knowledge at the right time, or the flaws might be too integrated in the design later on. Another critical issue is the attitude of the person learning; the learner's view of the material (important/relevant) and the educator (competent or not) are important. Kirkpatrick (1979) defines an evaluation matrix for training, in which he includes similar factors in the four levels:

learner reaction, learning, learner behaviour and organizational results. In our study we did not penetrate all these, due to resource limitations, but focused on the first three ones.

The learning level component is complex, as it cannot be measured quantitatively only. Birnbrauer (1987) describes the learning level measure of Kirkpatrick's evaluation model (1979) using the following components:

- knowledge measured by comparing what the person knows before and after the training.
- skills measured by testing the person's performance on new tasks.
- attitudes measured by evaluating the person's ability to cooperate and innovate, and her quality-mindedness.

In our study, we decided to include the elements *knowledge* and *skills* to some degree, but to exclude *attitudes*, since we did not perform the experiments in a full-scale, industrial environment (and due to limited resources). Of course, we are also interested in the user's perceived learning effects, but only on an anecdotal level.

Studying learning effects requires finding a way to relate the material presented and the person's ability to utilise that material after the training. In the case of learning design knowledge in the form of guidelines and style guides, this calls for an evaluation of the designer's ability to reuse the knowledge in later design situations. The major obstacle to this is the problem of relating different design situations to each other, in order to compare the knowledge used. We decided to test the knowledge transfer using a test task, in which the ability to evaluate and comment upon a completed design was measured. This way, we could adapt the source material (the guidelines and the design task) so that we could easily compare it with the performance in the test task.

A common method to study learning effects is to let the participants perform a test task before, and then again after the training session, which in our case was a design session. The obvious alternative is to use a control group and a training group, and compare their performance on a common after-training test task. We chose the latter approach, mainly because we considered the time required by the participants to be too long. In addition, designing test tasks that can be used before and after the design session, without creating equivalent tasks was considered rather difficult. It would be problematic to measure the *real* knowledge transfer using two design tasks; if they are too similar, participants may learn how to avoid problems

the second time. Moreover, they had to be similar, since we used a limited scope and design task. Hence, we had another argument for using a control group to detect knowledge transfer.

We decided to use an evaluation task as the post-training test, i.e., we let the participants evaluate user interface designs for the same design task as they themselves had worked on. In order to test for knowledge transfer (i.e., long-term memory), and not short-term memory effects, the post-training test had to take place after a certain amount of time, i.e., not immediately after the design session. We concluded that a 7–10 day delay would be sufficient.

The following specification was added to the experiment design:

7–10 days after the design session, a second, post-training, session is conducted. In this, the participants are given an evaluation task in order to measure knowledge transfer effects. The results of the control group are used for comparison.

3.5.5 Can interaction data be used to guide commenting behaviour?

As mentioned in the initial description of the research interests, we also wanted to explore algorithms for supporting the control of the commenting behaviour. We decided to explore the possibility of using raw interaction data (collected by logging user events in the design environment) for controlling commenting strategies. If a commenting tool is useful, and if an active mode is preferred, at least sometimes, then we believe that it is interesting to investigate the designer's perception of different situations where comments are delivered.

Probably, the designer considers some situations to be acceptable and others to be disturbing. If this is the case, it is reasonable to try to improve the delivery mechanism, making the designer perceive the situations as less disturbing. Hence, we wanted to explore potential correlation between the amount of user (inter)action and perceived MWL. A scenario and simple example, based on the assumption that the designer can only make use of comments if the “amount of interaction” is below a certain level, is shown in figure 3.2.

The rationale for this explorative study is straightforward; if an algorithm for improved comment delivery is required, then even a non-intuitive, complex, low-level mechanism, such as one based on user events, is valuable. Hence, we decided to collect interaction data during the design

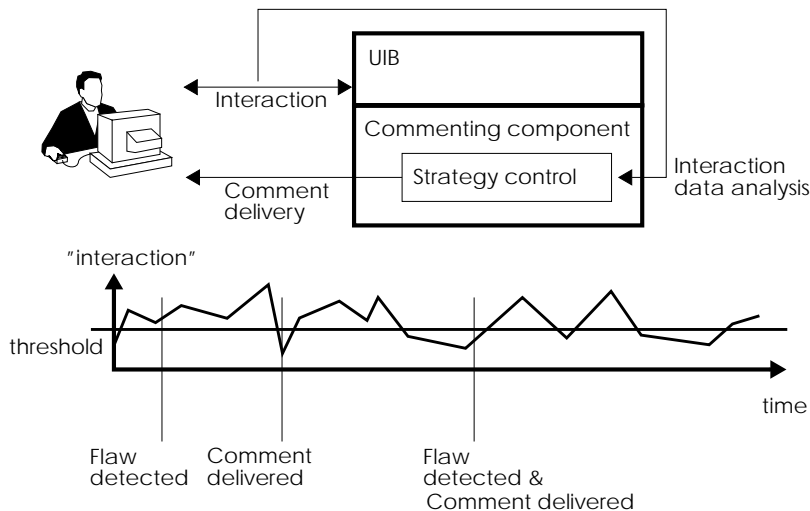


Figure 3.2 Scenario: using (low-level) interaction data to guide commenting behaviour

sessions, without trying to define events based on task-analysis or developing a scheme for interpreting different events. This part of the experiment did not in any way affect the participants, since it did neither affect the behaviour of the environment nor introduce any new intrusive data collection.

3.6 Experiment design

The research questions specified above resulted in a crossed design with two independent variables: active-passive mode and declarative-imperative mood. These were combined to generate four different conditions: active–declarative (a–d), active–imperative (a–i), passive–declarative (p–d) and passive–imperative (p–i). Since we wanted to evaluate the overall effects of a commenting system, we needed a control group (C). When describing differences between the control group and the rest of the groups, the latter are referred to as the S group (for “simulated system”). Figure 3.3 illustrates the design.

The dependent entities, usefulness, mood appropriateness, mode appropriateness and learning effects, were measured using a number of

different source variables, as described below. The elucidation of the research questions led to the following overall experiment design.

- We consider a single designer, working alone, using a graphical user interface design tool to implement a graphical user interface. The designer's task is specified in a written overview of the requirements. A set of user interface guidelines, more or less adapted for the task, is available in textual form.
- Two groups of participants are used: participants in group S use a complete design environment, with an integrated commenting system, while participants in a control group (C) use the design environment without the commenting component.
- The participants in the group using the commenting tool are divided into four different subgroups, in order to investigate the effects from using different commenting strategies. The four groups correspond to the combinations of the two independent dimensions: mood (imperative/declarative) and mode (active/passive).
- Immediately after the design session, each participant is given a questionnaire in which system usefulness and presentation appropriateness are measured. After this, a mental workload is measured using prompted recall. The designer is shown video sequences of design situations of interest, and then asked to fill out an RTLX form.
- To measure knowledge-transfer effects, a second, post-training, session is conducted 7–10 days after the design session. In this, the participants are given an evaluation task. The results of the control group are used for comparison.

	S-groups		Control group
	imperative	declarative	
active	a-i	a-d	C
passive	p-i	p-d	

Figure 3.3 Experiment design

3.6.1 Specifying the dependent variables

As described above, our experiment design required two different participant sessions: one where the participants designed an interface and one where they evaluated design solutions. The second session would be used

to collect data for the knowledge-transfer question; the first one would let us collect data for the other research questions.

We decided to use a questionnaire as the primary technique for collecting data, as explained above. In the questionnaire, we devised several sets of questions for each of the different dependent items. For each question, we decided to use a 8-point Likert type scale for the answer. We decided to use two measures for some questions: one *general* and one *specific*. The general measure would relate to the participant's perception of the total system and session, while the specific measure related to a single comment (see figure 3.4). This would allow us to reason about the overall features of the system as well as certain comment delivery behaviour.

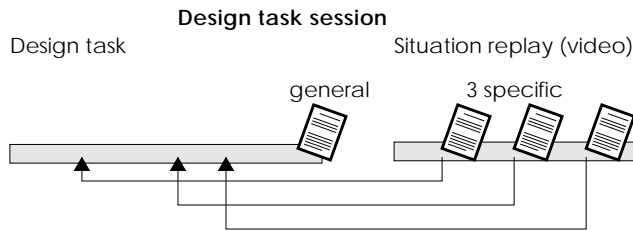


Figure 3.4 Ratings of the tool in general (questionnaire immediately following task) and its use in 2–3 specific situations (questionnaire after video replay).

In addition to the questionnaire, we considered knowledge transfer to be part of the system's usefulness. However, in this study, we did not include it as part of the analysis or in the conclusions.

As mentioned above, we also wanted to investigate the perceived MWL during the sessions using the commenting system. Although we believed that this kind of measure could give some information about the user's perception of the situation, we did not consider it to be reliable enough for using as part of the usefulness or appropriateness values. We chose to study MWL separately, for exploration only.

Usefulness

For the usefulness measure, we devised questions to check if the participant's impression of the commenting system was that it:

- was useful.
- helped them or hindered them.
- delivered new information.

- was something the participant wanted to use in the future.

By comparing the results of the S group with the control group, we would get data that described the differences in perceived usefulness. Hence, these questions should be given to all the participants, immediately after the design session.

Mode and mood appropriateness

For the mode appropriateness measure, we devised questions to check if the participant's impression of the commenting system was that it was:

- disturbing.
- supportive.

For the mood appropriateness measure, similar questions were devised, concerning the participant's impression of the commenting system. For this dimension, we wanted to check if their perception was that it:

- delivered comments that the participant could understand.
- was competent.

Of course, these questions concerned the S group only. To make it possible to relate to the delivery of comments, we decided to deliver these questions to the participants using the same technique as the one for MWL measurement described above.

Perceived MWL

Immediately after the design task, the experimenters selected a number of video sequences that contained interesting situations. To be able to compare effects of different strategies, the S group sequences must include several situations where comments are delivered. However, in order to investigate reliability and MWL differences, there must be a number of non-comment sequences as well, for comparison with the control group. We decided to use 2–3 sequences of approximately 1–2 minutes. Each sequence would be used for measuring both mode and mood appropriateness, and MWL.

Knowledge transfer

For the investigation of knowledge transfer, we devised a post-training task based on evaluation of user interfaces. In order to make it possible to relate evaluation performance and previous knowledge, the interfaces were based on the same task specification as the design task.

To investigate knowledge transfer effects, we would compare the performance of the S and control groups, as well as that of the different S groups. However, it would not be sufficient to compare the totals of each group; the comparison must somehow take into account the “quality” of the comments too. In order to balance for “erroneous” and “unimportant” comments by the participants, we decided to collect comments from a team of expert evaluators analysing the same set of interfaces. By asking the experts to report only those comments they considered useful, we would get a set of “useful” comments by which we could rate participant utterances.

Correlation between interaction data and subjective measures

For exploring the possibility of using interaction data (ID) to guide commenting strategies, we decided to investigate correlation between ID and the participants’ perception of specific situations, i.e. MWL and subjective ratings. In order to analyse the interaction logs, we developed software to decode and visualize the session data (as described in section 4.3), and to generate statistics for correlating and comparing interaction with other measures from the design session.

3.6.2 Data collection

Our design and the measures chosen above made us decide to use the following techniques for collecting data during the experiment:

- interaction logging and video recording for collecting data about the design session.
- a questionnaire for collecting demographic data, and data concerning the designer’s perception of the design session.
- a set of rating tasks for collecting data about the designer’s perceived MWL and perception of single comments.
- participant notes, video recording and experimenter observation for collecting data about the evaluation session.
- a post-experimental interview for collecting anecdotal data about the complete experiment.
- expert evaluator notes from the “useful comments” session

3.7 Pilots

Before the actual experiments, an informal pilot study was performed, in order to test and refine the procedure and material. Five university students

were asked to participate under the same conditions as in the actual experiment (see chapter 5). The five pilot participants were assigned to the five different condition groups and then performed the two tasks.

The input from the pilot sessions led to slight changes in the questionnaires, a more formalized interview session after the evaluation task, and a redesign of the UIB. The pilot sessions also served the important purpose of training the Wizard.

Chapter 4

The CODEK Software Environment

Our Wizard-of-Oz research approach called for a simulator of commenting tools, with special features for adaption of behaviour. We developed a simple system, adapted for Wizard-of-Oz experiments in the user interface design domain. This environment was used throughout the practical parts of the experiments in this study.

The system simulates a design environment including a user interface builder (UIB) and a commenting tool. From the user's point of view, the UIB is used to design a user-interface layout, while the commenting tool evaluates and comments upon the work. In another room, the Wizard observes the user's actions in a mirror environment, and enacts the behaviour of the commenting tool.

The system is designed with the intention to make it possible for the Wizard to see all the user's actions and modify everything the user has access to. The core of the system is a pair of UIBs, modified to communicate with each other. All the actions performed in the user's UIB are copied to the Wizard's tool, mirroring the design work to his screen. The commenting tool is implemented as a comment manager on the Wizard's side and a comment display on the user's side. Comments can be prepared and modified, and then sent to the comment display. The user may select and delete different texts in the display; all such actions are reflected in the Wizard's manager.

Designing the CODEK software environment was a non-trivial task, involving a series of important design decisions related to our research questions. Integrating those decisions with demands on the usability of the

tool, and at the same time having to make it ready fast resulted in some compromising and even “quick-and-dirty” solutions.

4.1 Simulating a design environment with a commenting component

Some of the system requirements were easy to identify and specify. The system should simulate a single user design tool with a commenting tool. Using the design tool should be very easy, in order not to add stress factors during the experiment. For this reason, simplicity, performance and a very short learning time were critical factors.

The types of design tasks to be used during the experiments were defined beforehand, hence some of the needs concerning widget set and functionality could be predicted and prepared for. It was decided that the system should run in a Unix environment on Sparc stations, since that kind of hardware could be easily accessed. We opted for the Motif look (OSF, 1991) when selecting software constraints, since that is a *de facto* standard in the Unix/X area.

With the limited time available, we decided to use ideas and influences from other UIBs and create a prototype quickly, instead of analysing the needs of designers and create an environment from scratch. By looking at prototypes and commercially available user interface builders, e.g., GINA (Berlage, 1991), Suit (Conway et al., 1992), TeleUSE (Alsys Inc., 1994) and Developer’s Guide (Sun Microsystems, Inc., 1991), we reached the conclusion that the following features was desired:

- overview of and easy access to the functions in the system.
- simple drag-and-drop creation of objects.
- a palette of predefined widgets.
- simple grouping/clustering of objects.

As for details related to the product, not much was required, since we were only interested in the static issues, i.e. the actual layout. Therefore, we did not need callback functionality or any other kind of dynamic mechanisms for the widgets. For the same reason, we decided that a hierarchical view of the objects in an interface, (tree view, as used in, e.g., TeleUSE and DevGuide) of the interface was not needed—the internal relationship between different objects was not important in this task. Furthermore, we believe that such a view would make the user perceive the tool as more complex than necessary for our purposes.

We had few but well defined initial design ideas regarding the commenting tool. They originated from combinations of observations of existing prototypes and our own requirements, resulting from the research questions. First, comments should be delivered as text. Secondly, we wanted some kind of mechanism that made it possible to associate comments with objects. Thirdly, the simulator should provide a simple mechanism for testing different moods and different behaviour.

The Wizard needed some sort of support environment for managing the actual simulation; monitoring the work, preparing and sending comments, etc. We studied reports on other Wizard-of-Oz environments, not only from the user interface design support domain, and applied the observations on our goals. The resulting demands on the simulator was:

- a single Wizard, single user environment.
- a database with canned texts for preparing comments rapidly.
- timesaving mechanisms for the Wizard, compensating the probable slowness of the human operator.
- a complete, modifiable, mirror of the user's design environment.

4.1.1 Platforms considered but rejected

Several approaches for developing the CODEK software environment were evaluated, but rejected in favour of the solution described above. Two of the main tracks analysed were:

Using a stand-alone builder, an X-multiplexer and a logging tool

The CODEK software environment includes a user interface builder, a large set of communication facilities and a logging application. As an alternative to integrating all these, thereby reducing the amount of effort required for communication technique, we analysed the possibility to use a multiplexer application in combination with stand-alone tools. This approach was considered impractical for the following reason: the commenting tool must be able to exchange data with the UIB (in order to associate comments with objects). Hence, the different components had to be able to exchange information, i.e., we needed access to the source code of all applications.

Using a commercial user interface builder

In order to create a more realistic environment and perhaps simplify the implementation effort, we analysed the possibility to take a commercially

available user interface tool and redesign it for our purposes. Two main problems made this path impractical: most commercial tools would have caused too much overhead for the participants (learning the environment), and the possibilities of getting access to source code for such an application were almost nonexistent in our project.

4.2 The simulated environment

The CODEK Wizard-of-Oz environment is a system for simulating and collecting data from the integration of user interface design support tools in a single-user context. It consists of two parts, the user environment and the Wizard environment, each one an integration of several subsystems. The two parts are linked to each other using a computer network. Figure 4.1 shows an overview of the system.

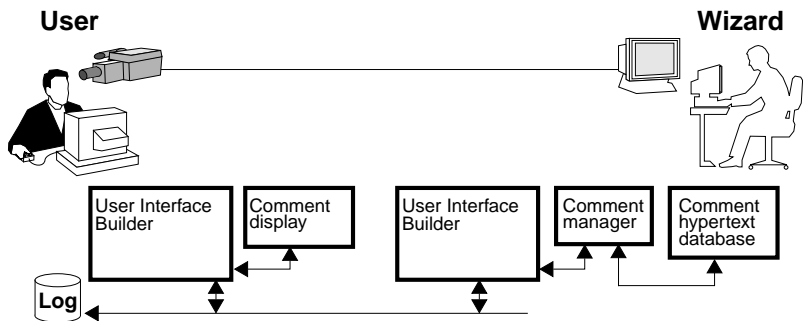


Figure 4.1 Overview of the CODEK Wizard-of-Oz environment

Besides the computer system, an audio/video link presents an image and the sound from the user's work environment on a display at the Wizard's workstation. This is to make it possible for the Wizard to better grasp the whole design situation.

4.2.1 The user's environment

From the user's point of view, the environment consists of a UIB and, depending on the experimental conditions, a commenting support tool. Figure 4.2 shows the two components as used in the experiments.

4.2.2 The user's UIB

The UIB is a rather simple user interface layout tool for designing the static parts of an interface. Using the UIB the designer is capable of creat-

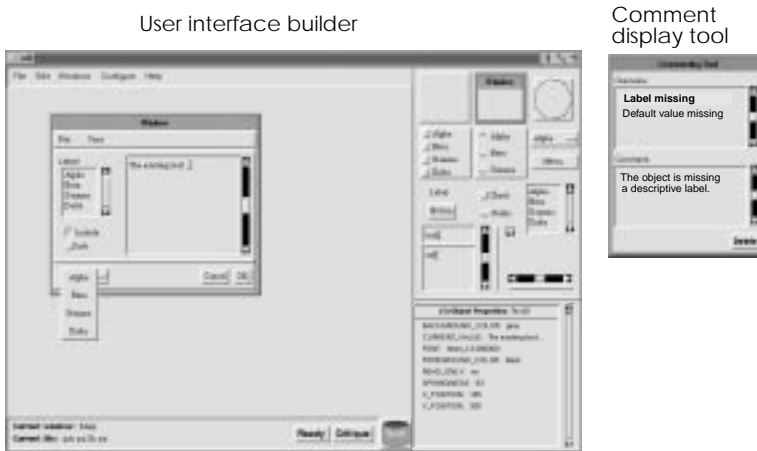


Figure 4.2 The user's screen environment

ing, modifying and testing a layout. The tool does not contain any mechanisms for supporting dynamic features or actual production of an interface, i.e. there are no modules for dialogue management, code generation or scripting.

A single window contains the three main parts of the UIB: the work area, the widget palette and the widget attribute editor. Creating an interface using the UIB is quite simple; objects are dragged from the palette, onto the work area. There they may be moved and resized using the mouse and edited using the attribute editor. The editor lets the user manipulate internal widget attributes, e.g. colour, text content and default values. Figure 4.3 shows the UIB window.

The UIB is intended for “standard” type interfaces, i.e., form based designs, and therefore provides a rather small set of widgets in the palette. The available objects are shown in figure 4.4.

In the attribute editor, a selection of attributes can be edited for each widget. Editing is easily initiated by clicking on an object. Since one of the main goals of the tool was simplicity, a maximum of ten attributes per widget is available, including only those considered important for our particular project.

All object creation, deletion and location actions are performed using the mouse. The user creates objects by dragging widgets from the palette to the work area. Deletion is done by dragging objects from the work area

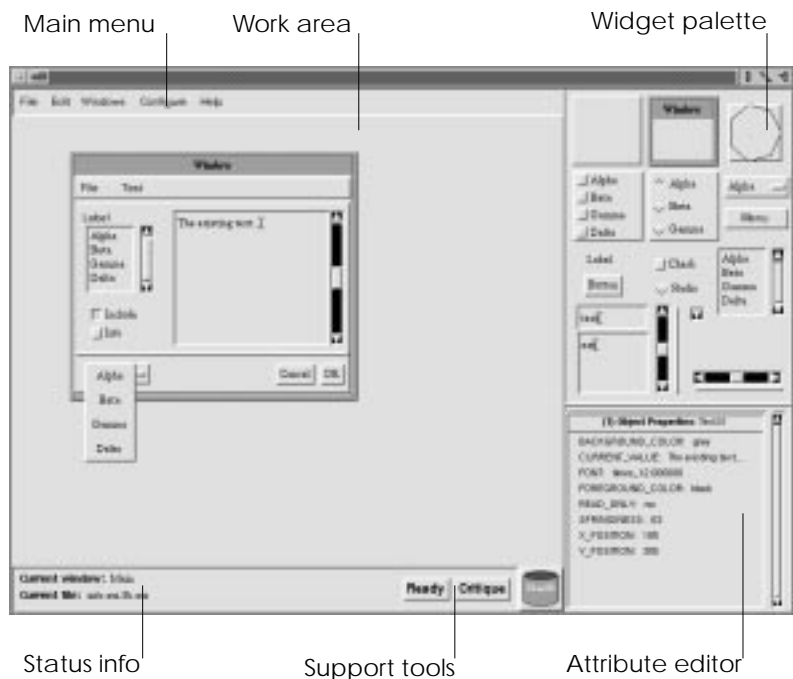


Figure 4.3 The UIB

to the trash can at the bottom of the screen. In order to move or resize an object, the user drags the object or its boundaries to the intended position.

In addition to the direct manipulation editing described, a set of layout functions is available via key commands or menu alternatives. The UIB provides the following functionality: *Copy*, *Paste*, *Delete*, *Group*, *Ungroup*, *Move to top*, *Move to bottom*.

Since the pilot studies indicated that the work area might not be sufficient for users creating multiple window solutions, a mechanism for handling several work areas was added. New areas can be created, and objects copied and moved between them. The user can still only see one area at the time.

4.2.3 The commenting tool display

For the commenting tool simulation sessions, a second window—the comment display tool—is added to the user's screen. This is where the actual

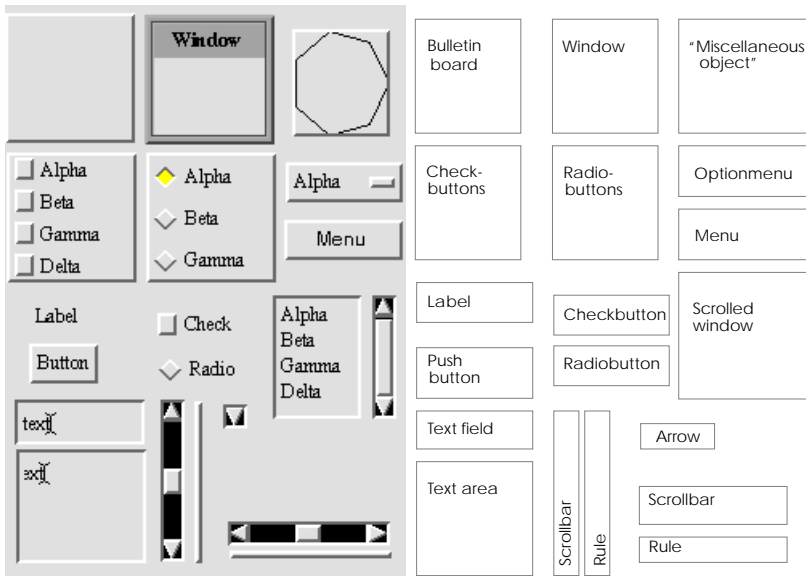


Figure 4.4 The objects available in the UIB

comments are displayed. A separate window is used in order to reduce the degree of interruption when the user is in the middle of a design step. Similar designs are found in prototypes by Reiterer (1994) and Stolze (1994).

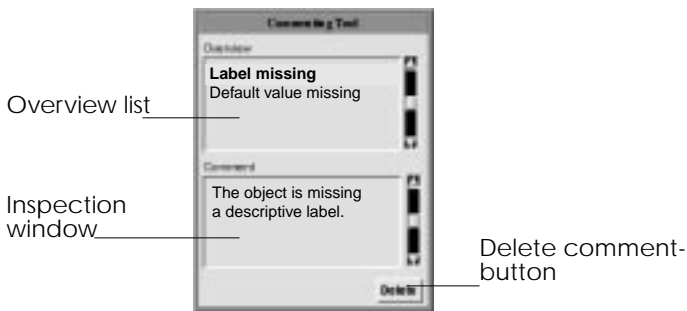


Figure 4.5 The commenting tool display

From the user's point of view, the commenting tool evaluates the layout represented in the work area and delivers comments in the display window. The tool is also used to handle the case were several comments apply,

and the user wants to be able to browse the received comment set. The commenting tool display is shown in figure 4.5.

Comment overview and inspection

A comment consists of two separate texts, a short overview phrase and a full comment sentence. The former is displayed in an overview list which displays the received comments, sorted by creation time; new comments are placed at the top of the list. By selecting an item in the overview list, the full comment text is displayed in a separate inspection area. In some cases it might be sufficient to read the short overview (“Add a label”), but in others, the full text may be necessary. Similar designs are found in prototypes by, e.g. Eriksson et al. (1996) and Harstad (1993).

At any time, the user may delete a comment from the system, by pressing the “delete” button at the bottom of the comment tool.

Objects associated with comments

Received comments may also be associated with certain objects in the work area, e.g. “The object is lacking a label”. In order to help the user keep track of the objects pointed out by the comments, an object highlight mechanism is built-in; associated objects are marked by a subdued flashing outline. The mechanism only applies for one comment at the time, the item selected in the overview list. By clicking on an item the involved objects and the comment’s full text are displayed. A similar solution was used in, e.g., Designer (Weitzman, 1992).

The highlight mechanism implies that new comments, if associated with objects, automatically try to catch the user’s attention. Other attentional cues were studied, but ruled out, considered to be too intrusive (e.g. sound effects, screen flashing or additional pointers).

4.2.4 The Wizard environment

The Wizard uses several tools to manage the Wizard-of-Oz experiments: a UIB, a comment hypertext database and a comment manager. To produce comments and present them to the user, the Wizard retrieves texts from the hypertext database, edits them, associates them with objects in the work area, and then submits them to the comment manager. From there, he can send them to the user’s display tool. The Wizard also has a communication window, in which he can conduct a limited dialogue with the experimenter in the lab room. Figure 4.6 shows an overview of the Wizard’s environment.

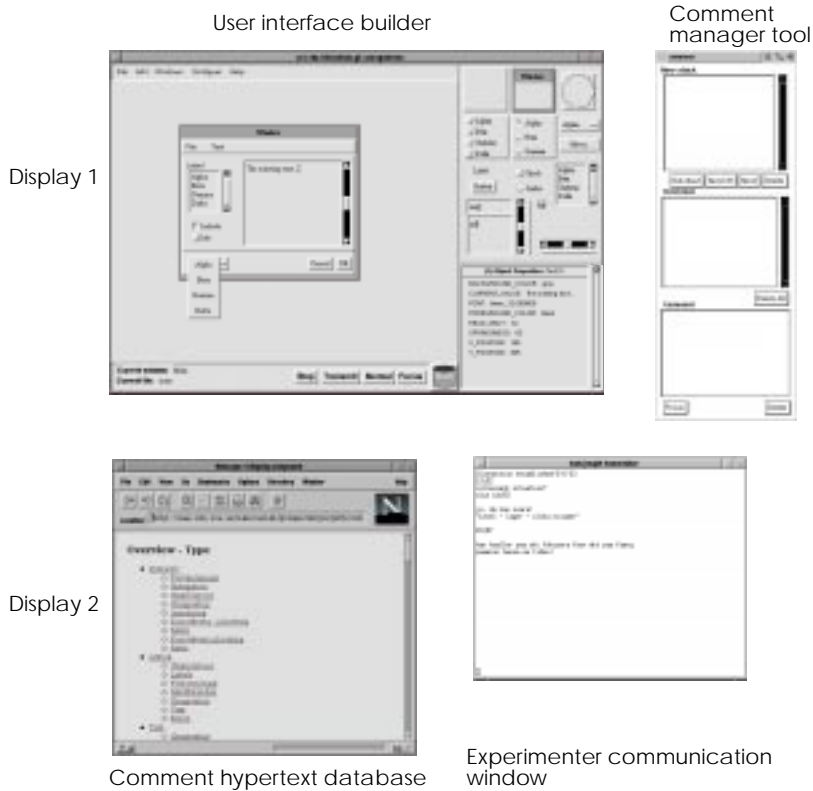


Figure 4.6 The Wizard's screen environment: display 2 was located to the left of display 1.

4.2.5 The Wizard's UIB

The UIB used by the Wizard resembles all the capabilities of the user's UIB, plus a set of functions for managing Wizard-of-Oz experiments. To make it possible to interpret the user's work, the tool displays any action performed by the user, e.g. creating objects, setting attributes, resizing or moving widgets. Actually, the mirroring of actions works the other way too, making it possible to test active design agent technology, although that wasn't utilized here. For the CODEK experiments, the Wizard UIB was used primarily to watch and prepare comments.

Most of the added functionality is intertwined with the comment manager, and is described there. However, some extra mechanisms are specific to the Wizard's UIB (see figure 4.7):

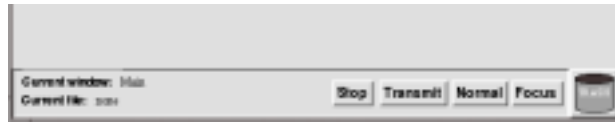


Figure 4.7 Extra controls in the Wizard's UIB

Stop/Restart

In cases where the Wizard needs time, i.e. requires the user to wait, he can lock the user's environment by disabling all the functions and changing the cursor into a small "busy clock" icon. This kind of mechanism may be necessary in Wizard-of-Oz experiments, as a human often performs considerably slower than the intended system. The inclusion of this function was inspired by the similar use of waiting cues ("Please wait", "The system is working...") in the Wizard-of-Oz environments by Dahlbäck and Jönsson (1992, 1993). Those cues proved to be very useful, making longer response times easier to accept.

Manual focus/unfocus

For future work, the UIB has a mechanism for manually making an object focused and later on, unfocused.

Transmit

In case of a breakdown, crash etc., if the user's UIB has been restarted, the Wizard has a function that can transmit all the contents of his UIB to the user's UIB in order to save time.

4.2.6 The comment hypertext database

The database of comments was accessed via a hypertext interface, using a common WWW browser. The Wizard browses the database, using only the overview phrases as navigational help; the comments are arranged and grouped by conceptual level and subject. After having selected a suitable comment, the full text is displayed.

When the comment text is prepared, the Wizard may choose to associate the comment with some objects in the work area. This is done by first selecting the objects in the UIB, and then "sending" the prepared comment

to the comment manager's "new" buffer. Comments in the buffer are only visible for the Wizard.

The hypertext system consisted of a WWW browser (Netscape Navigator), for navigating and selecting guidelines; a WWW server (Luotonen et al., 1996a) where the information was stored and fetched and a CGI (Common Gateway Interface, see Luotonen et al., 1996b) script for handling the communication with the rest of the CODEK system.

4.2.7 The comment manager

The Wizard's comment manager is an extended version of the user's comment display tool. It contains an overview list and an inspection area, and in addition also a "new" buffer for preparing comments before sending them to the user. The buffer is useful in situations where the Wizard is able to predict future violations.

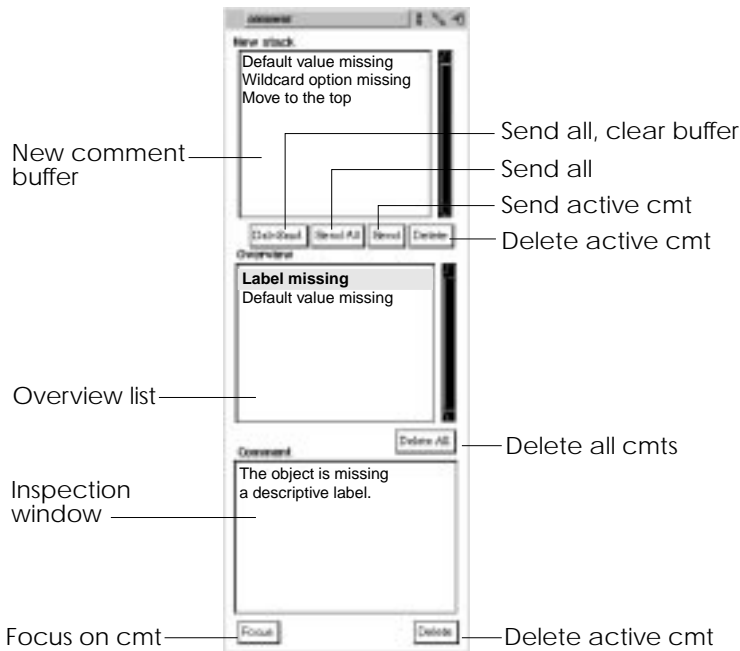


Figure 4.8 The comment manager

The overview list shows an exact copy of the user's overview list. Any actions performed by the user are mirrored to the Wizard's list, i.e. selecting an item or deletion. Of course, this affects the inspection window, and involved objects too. Unlike the work area two-way communication, the Wizard's list is not mirrored to the user's tool as well. This allows the Wizard to inspect delivered comments without affecting the user's display tool.

New comments, entered via the hypertext database, are placed in the buffer, displayed using the overview phrase only. Like the overview list, the comments are sorted by creation time. The Wizard may decide to send one comment at the time or all at once, using the buttons "Send" or "Send All"; this moves the comment(s) to the overview list in both the Wizard's and the user's tools. There is also a "Delete" button to make it possible to remove a prepared comment that becomes invalid. By selecting a comment in the buffer, the associated objects are highlighted in the Wizard's work area, overriding the ordinary inspection mechanism.

The comment manager also has a set of functions that helps managing already delivered comments. A "Delete All" button makes it possible to remove all the delivered comments. By pressing the "Focus" button, the Wizard can force the selected comment to become focused. Hence the Wizard may override selection actions performed by the user, if necessary (although this was not utilized in our experiment).

4.2.8 The complete Wizard-of-Oz environment

The modules and subsystems described in the previous sections together form the complete Wizard-of-Oz system used in the experiments. Figure 4.9 shows how the different parts are connected to each other via sockets. The rationale behind the current architecture is described below.

The two main components—the UIB tools—are used to send information back and forth between the Wizard and the participant. On each side the comment tools are connected to the UIB, making all communication related to comments pass through the UIBs.

It might seem awkward to implement the different modules as separate applications, and then solve communication via sockets, instead of just letting the comment tool(s) be internal modules, with separate windows, of the UIB application. However, this was not possible using the interface package we selected (SUIT) in which it is only possible to have one window per application.

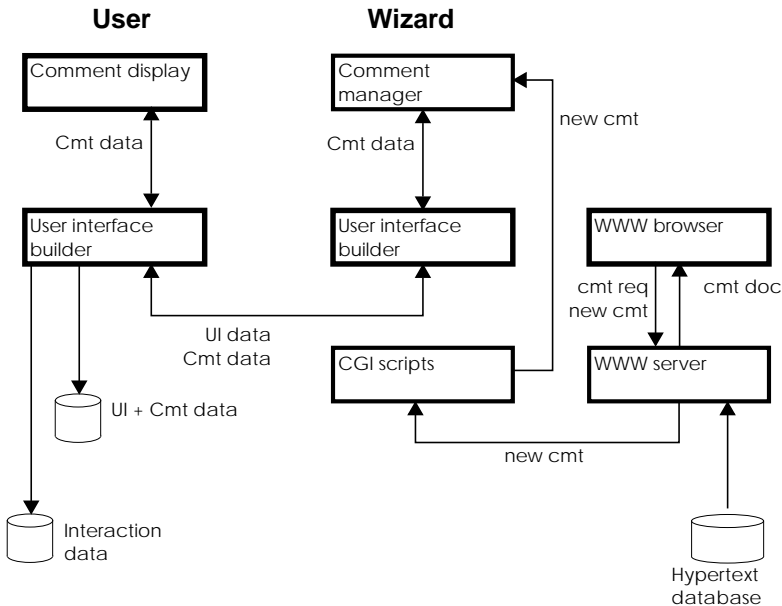


Figure 4.9 The complete Wizard-of-Oz system

Interaction logging

The Wizard-of-Oz environment is equipped with a logging mechanism, recording interaction and communication data to files. The purpose of the logging is twofold: it makes it possible to replay user interaction and prepares for statistical analysis of the users interaction. Two separate log files are created by the environment: one containing user interaction data and one containing communication data.

The interaction log is action-focused, recording only completed actions, e.g. *object moved*, *group created*. This means that we do not capture low-level events such as mouse movement or single keyboard taps. The data in the interaction log includes the following type of information:

- user actions (UIB).
- Wizard actions (UIB).
- comment actions (comment tool components).

Each item (recorded action) contains information about:

- action type.
- start and stop time.

- objects involved.
- origin (user, Wizard).

In addition to the user interaction log, the communication between the different subsystems of the environment is also written to a log file. This log becomes much larger, and contains all the information necessary to replay (parts of) a session, i.e., user interface changes and comment actions.

4.3 Low-level issues

The CODEK environment was developed and used on a Unix platform. The software, at present a total of 50 KLOC, was coded mainly in C and the interface parts were created with the toolkit SUIT (Conway et al., 1992), using the Motif look. Some Perl code was used for the component communication.

Two workstations were used for the experiments—the Wizard had a SPARC 5 and the user a SPARC 20, both with a standard keyboard (Swedish mapping) and a Sun optical mouse. The computers were equipped with two 19" colour screens each.

4.3.1 UIB interface representation

The interfaces developed in the CODEK environment were managed internally using the SUIT toolkit's mechanisms, i.e. the internal representation concerned layout only. This implies that there was no higher-level representation or connection to domain-specific information internally. However, representing and managing such information is technically possible (see section 2.6).

The external representation, i.e. the files used to store layouts were also based on the SUIT standard representation, as depicted in figure 4.10.

```
MAKE("PushB4","button","ROOT",0);
SET("PushB4","background color","GP_color",0,0,"grey,
white");
SET("PushB4","font","GP_font",0,0,"times",12.000000);
SET("PushB4","foreground color","GP_color",0,0,"black,
black");
SET("PushB4","label","text",0,0,"Button");
SET("PushB4","viewport","viewport",0,0,"653 361 697 385");
```

Figure 4.10 The external representation of an interface object in the CODEK environment. Example: a push button.

4.3.2 Shared comment and interface representation

All information that was shared between the Wizard and the user was stored in both of the tools, i.e. the data was not physically shared, but kept in sync in two different applications. Every change in one tool resulted in a message being sent to the other side, causing the corresponding representation to update. The built-in mechanisms for handling the interface representation as textual descriptions made it possible to interchange layout descriptions easily.

The delivered comments were handled in a structure as depicted in figure 4.11. Each comment is identified by its guideline number and consists of an overview text and a full textual description. Each comment may be associated with one or more objects in the work area. The system also allows an object to be associated with several comments.

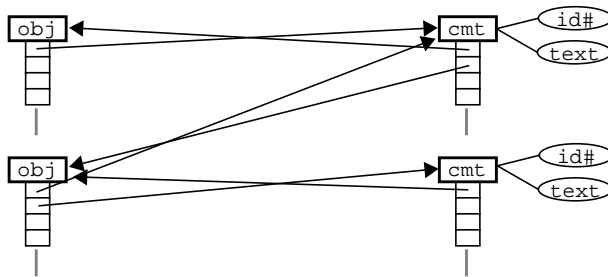


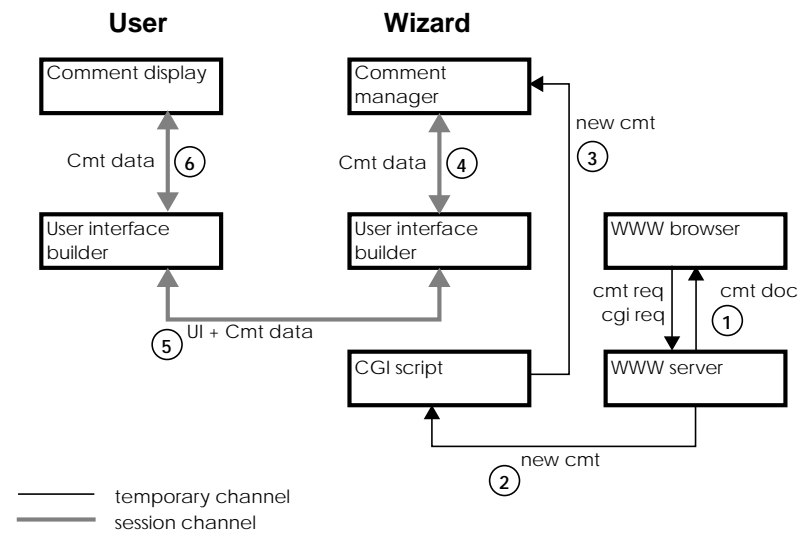
Figure 4.11 The data structure for handling delivered comments

The bidirectional links between comments and objects are used to handle situations where a comment or an object are to be deleted. In our experiment setup, removal of an object automatically removes any associated comment (link from object to comment). The links from comments to objects are used for the object highlight mechanism when a comment is selected.

4.3.3 Communication

The Wizard-user application communication was handled using standard Unix sockets, as depicted in figure 4.12. The approach used here was slightly cumbersome in that we communicated “through” the UIB applications from the “outer” applications, i.e. the commenting parts. The reasons for doing this was partly because of the limitations of the SUIT package,

as described above (section 4.2.8), and partly in order to simplify the logging mechanism—all logging was performed by mechanisms in the UIBs.



#	Channel	Data
1	http connections between WWW browser and server	Requests for cmt documents Requests for adding comments Comment documents from server
2	CGI requests from server to script	Requests for adding comments: comment id + content
3	CGI-script – Comment manager	Requests for adding comments: comment id + content
4	Comment manager – UIB	Comment actions: add, delete, focus
5	UIB – UIB	UI actions: create, move, size,... Comment actions: add, delete, focus
6	UIB – Comment display	Comment actions: add, delete, focus

Figure 4.12 The different communication channels used in the CODEK environment

A simple, custom protocol was designed for handling the communication. In this, we relied on the performance of the workstations and the net-

work, not including handshaking or acknowledge messages unless absolutely necessary. The data transmitted was kept in textual form the whole time, and the interface object information could therefore be sent exactly in the form used for the external representation (see section 4.3.1). A possible drawback of this is that large compound objects (groups) or complete interfaces may delay the system, and in extreme cases possibly result in lost data (due to limitations of the hardware and socket mechanisms). However, for our purposes the capacity of the system was sufficient.

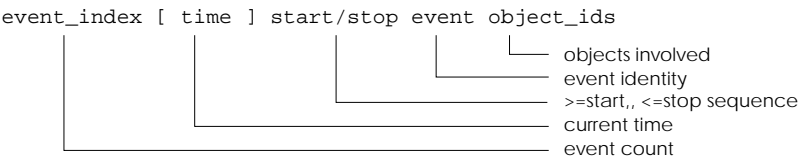
The communication between the hypertext database and the Wizard's comment tool was also solved using sockets, via a Common Gateway Interface (CGI) script on the WWW server (Luotonen et al., 1996a; Luotonen et al., 1996b). The server was run on the same machine as the Wizard's system, in order to improve performance. When the Wizard selected a guideline for inclusion in the comment tool, the content of the guideline display in the browser was posted to the CGI script. The script then passed the information, via a socket, to the Wizard's comment tool, where the text was pushed into the buffer.

4.3.4 Interaction logging

The interaction and communication logging is hard-coded in the application. For every higher-level action and communications function available in the environment, a logging hook has been added. As was described above, all the logging is controlled by a module in the user's UIB; all action and event data is sent to this module from all other subsystems. The interaction log contents is illustrated in figure 4.13.

In order to make it simple to interpret and use the communication log files, they were designed to contain data on the same form as the client-server communication data (although compressed and coded). Hence they look a lot like the SUIT standard external representation format. Small modifications to the SUIT package therefore made it possible to create a playback mechanism.

Log record line

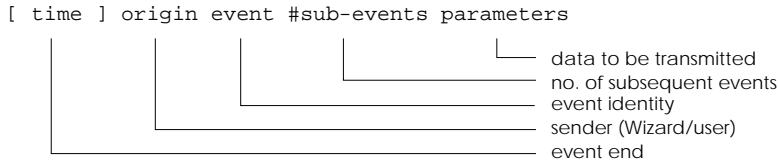


Example

```
32: [ 0 3 21 6] > drag
33: [ 0 3 26 706] create Push Button28
34: [ 0 3 26 719] move Push Button28 viewport
35: [ 0 3 26 728] < drag
36: [ 0 3 28 63] edit Push Button28
37: [ 0 3 38 206] > set property
38: [ 0 4 2 686] setprop Push Button28 label
39: [ 0 4 2 735] < set property
```

Figure 4.13 Contents of the interaction log file

Log record line



Example

```
[0 16 55 971] 0 2 0 OptionB42;viewport;"150 378 222 402"
[0 16 57 222] 1 13 0 TextField38@@TextField41
[0 1 33 97] 0 0 3 Label28;10
[0 1 33 102] 0 20 0 label;"Label"
[0 1 33 105] 0 20 0 viewport;"227 374 263 398"
[0 1 33 100] 0 20 0 font;"times,,12.000000"
```

Figure 4.14 Contents of the communication log file

Chapter 5

The Empirical Study

The experimental data was collected during a period of three months in the spring of 1995. Each participant attended two separate sessions, the design session and the evaluation session, as depicted by the schedule in figure 5.1. The participants were divided into five groups, one for each experimental condition and a control group.

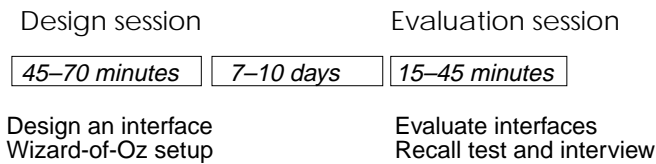


Figure 5.1 The experiment sessions

The design session included practical work with a design task, a short questionnaire and MWL assessment. During the practical work, the participants used the UIB to design the static parts of a user interface for a hotel search and reservation system. The control group used only the UIB; the other groups also had a simulated support tool capable of evaluating and commenting upon the design. The evaluation session took place 7–10 days after the design session. At this occasion, the participants reviewed and commented upon a set of different design proposals for the design task.

5.1 Experimental design

A crossed design, consisting of five groups, was used for the experiment. Four groups were used to investigate the effects of different commenting strategies. The two appropriateness between-subjects factors were the

mode (active or passive) and mood (imperative or declarative) of the simulated commenting system. An additional control group was used to study usefulness, yielding the basic between-subjects factor: commenting support (yes/no).

5.1.1 Experiment participants

A total of 16 university students participated in the study, 15 male and 1 female. All were experienced computer users, 11 were students of the Computer Science programmes and 5 were students of other programmes at the university. All had at least some experience with HCI and user interface design, i.e., at least one course in HCI and cognitive science and/or experience from using user interface design tools in small-size systems development projects.

Most of the participants volunteered in response to an invitation distributed by us. The invitation described the experiment as a study of the usefulness of commenting expert system support in user interface design. In return, each participant was rewarded with a movie ticket (approx. \$10).

The experiment description mentioned that each person was to be recorded on video and that a total of 2 hours plus half an hour one week later was required. Nothing was said about the Wizard-of-Oz approach. The ones actively solicited by us were given the same written description. The participants were allowed to select date and time for the two sessions themselves from a schedule.

Upon signing up for the experiment, the participants were randomly assigned to one of the five groups of different conditions. This resulted in four (4) participants for the a-i conditions and three (3) in each of the other groups: a-d, p-i, p-d and control.

5.2 Written material

Three sets of written material was developed for the experiment: a design task specification, questionnaires and RTLX forms for collecting data, design proposals for the evaluation task, and instructions for the expert evaluators. All the material was tested and some of it redesigned slightly during the pilot sessions.

The design task specification

The design task was presented in a written specification, see appendix A. The overall goal was described as “designing and implementing the visual

parts of a hotel search and reservation system". The specification contained information about:

- the goal and intended use of the system.
- requirements concerning data fields and functionality.

The specification was written in a form adapted for the goals of the research project and the current environment. Some issues of importance and constraints on the contents were, e.g.:

- Make the specification "realistic", i.e., the contents should be similar to the potential deliverables from the work before detailed design.
- Minimize the time needed for systems and task analysis by the participants, and make it possible for them to start and perform the detailed design of the specified user interface.
- Make the specification match the design environment, i.e., the widgets and the functionality for modifying those.
- Make the task solvable within 30–60 minutes.

Questionnaire and RTLX forms

Three sets of questionnaires and forms were used for collecting data during the design task.

Background information

A set of basic questions concerning the participant's age, background, education, etc. This questionnaire was delivered to all participants.

Perception of the total environment in general

Nine rating scales addressing the perception of the total environment. This questionnaire was delivered to all participants. See appendix B.1.

Perception of the commenting system in general

Eight rating scales addressing the perception of the commenting tool only. This questionnaire was delivered to the S groups only. See appendix B.2.

Perceived MWL in specific situations

A TLX form for measuring the participant's perceived MWL during 2–3 different situations (replayed from video). Two to three copies of this form were delivered to all participants. See appendix C.

Perception of specific commenting situations

Seven rating scales addressing the perception of a situation where a comment has been delivered. This questionnaire was delivered to the S groups only, at the same time as the TLX forms. See appendix B.3.

The evaluation proposals

For the evaluation sessions, three design proposals for the design task were prepared and printed on paper. The proposals, labelled 1–3, are depicted in appendix D. The three proposals were generated by the experimenters, using the following guidelines:

- they must be based on the specification used in the design task.
- three different basic layout approaches (e.g. single/multiple window solutions) should be used.

In each proposal, ten detail design flaws had been planted by the experimenters. The flaws were generated by creating a design solution that violated a guideline from the knowledge-base used during the design session. Of course, the design proposals may also contain other flaws, not covered by the knowledge-base guidelines.

Expert evaluator instructions

A written instruction was prepared for the expert evaluation of the three design proposals. The instructions includes a set of guidelines for the evaluation task (see appendix D.1):

- apply useful comments only.
- work at most 30 minutes on the evaluation.
- register/record each comment using both a textual description and a mark in the corresponding interface proposal.

In addition to the instructions, the experts were given the written design task specification (see above), and a short overview of the UIB.

5.3 The CODEK experiment environment

Two rooms were used for the experiments, one for the participant (the experiment room) and one for the Wizard (the Wizard room), both at the Computer Science Department. The rooms were located some 10 meters from each other, which made it impossible for the participant to notice the actual settings, but possible for the experimenters to meet outside, if necessary.

5.3.1 The experiment room

The experiment room was furnished as depicted in figure 5.2. During the design task, the participant sat at a desk in front of a SUN 20 workstation, with the possibility to adjust the height, level and distance of the monitor and keyboard if necessary. Next to the participant, the experimenter was sitting at a similar desk. A video camera was set up to record the screen of the participant and the conversation between the participant and the experimenter.

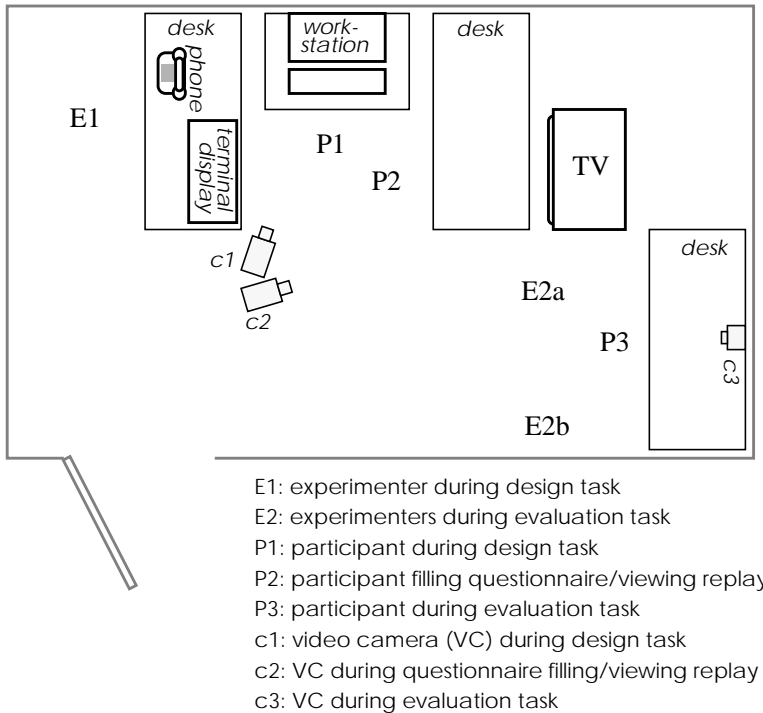


Figure 5.2 The experiment room and the equipment setup

At the experimenter's desk, a second screen was connected to the workstation, turned away from the participant. This screen was used to display messages from the Wizard, if necessary. The experimenter also had a fake phone at his desk, used only to call the other experimenter (the Wizard) when the design task was finished. This made the sudden arrival (at the end of the session) of the other experimenter plausible.

For the MWL rating sessions after the design task, a large TV monitor was used to replay the recordings from the participant's work. While watching this, the participant had access to a large desk for answering the questionnaires and the rating questions.

During the evaluation task, the participant sat at another large desk in the experiment room. The two experimenters sat behind the participant, in order to be able to study the evaluation work. A video camera was set up recording the desk from above.

5.3.2 The Wizard's room

The Wizard's environment consisted of a SUN 5 workstation with dual screens (19" colour), plus a small video display monitor (see figure 5.3). The right screen was used for the UIB and the comment manager. On the left screen, the Wizard accessed the hypertext database, monitored the user-Wizard communication and sent messages to the other experimenter's screen.

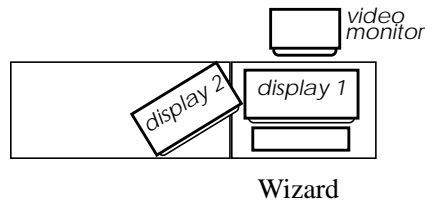


Figure 5.3 The Wizard's work environment and equipment setup.

Above the two screens, a small monitor displayed the image captured by the video camera in the experiment room. The sound recorded in the experiment room was monitored via a set of headphones. All this gave the Wizard visual and auditory access to the experiment room.

5.3.3 The computer environment

Two workstations were used for the experiments—the Wizard had a SPARC 5 and the user a SPARC 20, both with a standard keyboard (Swedish mapping) and a Sun optical mouse. The computers were equipped with two 19" colour screens each.

The CODEK environment was developed and used on a Unix platform. The visible parts of the software was created primarily with the toolkit SUIT (Conway et al., 1992), using the Motif (OSF, 1991) look. All textual interface components were in English, mainly because this is often the

language used in commercial development tools, and participants were all familiar with at least one such tool. All feedback from the commenting system, i.e., the comments, were in Swedish, since we did not want to bother the participants with translation of guidelines.

5.3.4 Video and audio equipment

One portable Hi-8 video camera, a Sony CCD-V800E, was used for recording the experiment. The camera was adjusted to permit flicker-free recording of the workstation colour screen. The camera's built-in microphone was used to record comments from the participant during the sessions. Three different locations and aimpoints were used during the experiment (see figure 5.2):

- During the design task, it was aimed at workstation's screen, located behind the participant. Only the screen was visible in the picture.
- During the subjective rating and perceived MWL measurement, it was aimed at the video sequence replay monitor (the large TV monitor), located behind the participant. Only the replay monitor and the experimenters were visible in the picture.
- During the evaluation session, it was aimed at the desk where the evaluation took place, located above the desk. Only the printed design proposals and the participant's hands were visible in the picture.

Two cassettes, one Hi-8 and one standard VHS, were produced during each session. The video camera was connected to a VHS video recorder where a standard VHS video cassette recorded the material. The TV monitor used to replay video-sequences after the design task was a 26" colour TV connected to the video recorder. The video recorder was also connected to the 14" monochrome monitor in the Wizard's room, making it possible to monitor the design task. The audio signal was transferred from the video recorder to a pair of headphones used by the Wizard.

5.4 The Wizard and the guidelines

In any Wizard-of-Oz experiment, the Wizard becomes a most important factor and potential source of bias and/or noise. The results of a Wizard-of-Oz experiment must always be interpreted with the Wizard characteristics in mind; the kind of knowledge used, response strategies, breakdown strategies, etc.

The CODEK Wizard used a predefined knowledge-base, which was the primary action guide, at the same time as it defined the vocabulary of

the system. A set of instructions added further guidance, and also restricted the Wizard’s behaviour for the purpose of our experiment. However, we disregarded the technical feasibility of the simulated tool to some extent, since we were mainly interested in the mood and mode dimensions. Therefore, the performance of the resulting system may in some cases be “unrealistic”, compared to state-of-the-art tools of today.

5.4.1 Knowledge and context

The Wizard used a knowledge-base of 33 different guidelines, each rule available in the two moods: declarative and imperative. Two thirds of the guidelines were lexical and syntactic, adapted from well-known sets of user-interface design guidelines (Brown, 1988, Smith and Mosier, 1986, NASA, 1992). One third was based on domain and task knowledge for the specific task, including the specification (see appendix A.1). Table 5.1 illustrates the different knowledge levels and the distinction between declarative and imperative mood

	Declarative	Imperative
Lexical	The object is lacking a descriptive label.	Provide a descriptive label for the object.
Syntactic	The object is lacking a default value.	Provide a default value for the object.
Task	The object(s) are not arranged to support the solving of the task.	Arrange the object(s) to support the solving of the task.
Specification/ Domain	An overview of the search results is missing.	Include an overview of the search results.

Table 5.1 Examples of comments in the hypertext database

The selection of guidelines was made by the experimenters, guided by the task specification and design proposals. In order to filter out (and investigate the request for alternative guidelines) the knowledge-base was tested in two steps: 1) informally by the experimenters, applying them on the design task specification and the design proposals, and 2) during the pilot studies.

All the rules were formulated as textual comments, and were stored in a hypertext database. The Wizard had access to them via a hypertext structure, as depicted in figure 5.4. Three different overview views of the comments were available, sorted by level of content (semantic, syntactic, etc.),

domain (label, window, colour, etc.) and subject (i.e., alphabetically by overview title).

In the display of an actual guideline, the Wizard had the possibility to edit the textual contents. However, this was never used during the CODEK experiments.

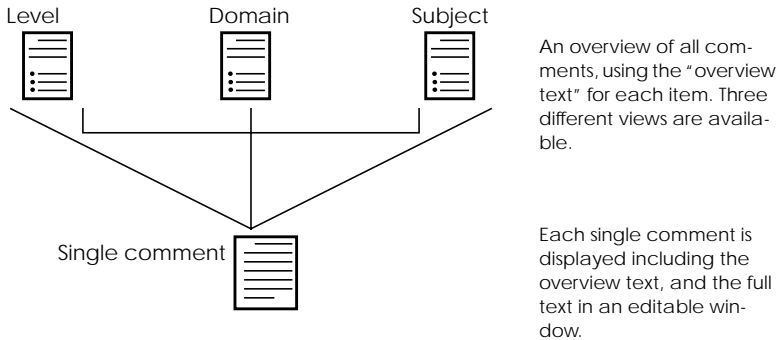


Figure 5.4 The hypertext structure used by the Wizard to access the guideline database

“External knowledge”

Besides the knowledge-base, the Wizard had access to information about the participant’s work and situation. The purpose of this was to enhance the Wizard’s performance, by giving him enough information to “predict” future design flaws. The following audiovisual information channels were available at the Wizard’s workplace:

- a mirror display of the user’s UIB.
- a video link, showing the user and his screen from behind.
- an audio link, recording the sound in the lab.

The Wizard also had access to some non-direct background information (his own experience and memory):

- the rationale behind the original guidelines used for the knowledge-base.
- his previous knowledge about user interface design.
- ideas of possible implementations of the user’s task, from earlier phases of the research project.

5.4.2 Wizard work instructions

In order to assure that the simulated environment “behaved” in a similar manner through all the sessions, the Wizard had a set of instructions that he followed strictly. The main point here is that he consistently strived to deliver all applicable comments.

The instructions were rehearsed and tested during a set of informal sessions run within the project team, and during the pilot sessions.

Overall

The overall instructions used by the Wizard were:

- deliver all applicable comments.
- prepare comments as soon as a flaw is visible for the Wizard, hence also by the participant.
- do not deliver comments in advance, i.e., the simulated system shall not be preventive in any way.
- use the display mirror, user video and audio data in order to identify guideline violations.
- do not deliver a comment about an object/group of objects that the user is currently working on (i.e. an object in focus, active in the attribute editor, etc.).
- if a comment is associated with an object that the user deletes, the comment is automatically deleted too, in order not to confuse the user by keeping a comment pointing out an object that no longer exists.
- try to “predict” problematic situations/guideline violations by looking at the copy of the participant’s screen, the monitor and listening to the user via the headphones. Store “potential” comments in the buffer.
- if the user has an opinion/question about the capabilities of the system or interpretations of the requirements that he/she expresses to the experimenter, but that the experimenter and Wizard-of-Oz has not discussed beforehand, the Wizard-of-Oz decides how the experimenter should answer, by sending a textual message to the experimenter.

The Wizard also had a set of specific instructions for the two different settings active and passive. This was due to the fact that the term “active” had to be specified in terms of what kinds of actions were to be affected, e.g. deletion of comments.

Active

1. whenever a comment is applicable: fetch and deliver it immediately

2. as soon as a delivered comment is no longer applicable (deleted objects, guideline violation rectified by user): delete the comment from the user's list (if it is not already deleted by the user)

Passive

1. whenever a comment is applicable: fetch and store it in the buffer
2. as soon as the user asks for comments: deliver the applicable comments stored in the buffer, and delete comments no longer applicable from the user's list.

5.4.3 “Unrealistic features”

The resulting simulated system embodies some features not easily implemented using today's techniques and algorithms. However, as we explain in chapter 3, the main objective is *not* to simulate an existing system, but some of the characteristics of a possibly optimal system. Hence, features that are computationally hard today, or features that result from a human's ability to “predict” in order to improve performance are not major problems. The possible drawbacks of this approach are discussed in chapter 8. Features not easily handled today include:

Domain and task knowledge

The simulated system “knows” about the design task and has knowledge about the domain. Most current prototype design environments do not handle such information.

User situation assessment

The simulated system has access to a complex image of the behaviour of the designer, via the interaction mirror, video and audio links. A design environment would typically have access to interaction and current status data only.

Prediction of future guideline violations

The Wizard, being a human, and also having as a primary goal to deliver all comments applicable, has the human ability to predict and guess future flaws in the designers work. However, the only possible improvement from this is a faster system, since comments are only delivered after the actual appearance of a flaw.

5.5 Procedure

The experimental data was collected during two separate sessions for each participant, the design task and the evaluation task.

5.5.1 The design task

As the participants entered the experiment room, they were seated at the computer desk and given an overall description of the experiment. They were told that they were to be given a design task, after which a short interview would be held. As indicated by the information in the invitation, they were informed that the design work was to be carried out in a prototype design environment.

Some introductory questions about their knowledge of user interfaces and design tools were asked in order to prepare for the next step. One experimenter was present during the entire session, seated at another computer desk beside the participant (see figure 5.2).

Every participant was given a short (approx. 15 minutes) guided tour of the design environment they were to use in the task. The tour, held by one of the experimenters, covered an introduction of the UIB and some terminology, depending on the participant's experience. A complete walk-through of all menus, functions and objects on the widget palette was given to all participants.

All except the control group participants were given a description of the commenting tool, with different contents depending on the experimental condition to be used (active/passive). The active group was told that they were going to receive comments from a built-in commenting component automatically during the work. The passive group was told that there was a commenting component built-in, and that they could request comments at any time during the work.

The participants were then given the written task specification, and a quick-reference sheet for the UIB. They were informed that they were allowed to work for at most one hour solving the task, which included designing and implementing the static parts only of the hotel search and reservation system. The experimenters reminded those having worked more than 50 minutes about the time limit, and asked those participants to start wrapping up.

Throughout the experiment, the participants were allowed to ask questions concerning the design environment and interpretations of the written requirements. No questions about or discussions concerning design solu-

tions or work strategies were allowed. The present experimenter answered the questions after having consulted previous answers, and comments from the Wizard (on the message monitor). Both experimenters created a free-form protocol with time-stamps of the session, recording annotations about interesting sequences, problematic actions, comments and breakdowns.

When the participants had finished the design task, they filled out the questionnaire covering background data and ratings of their perception of the task and the design environment (see appendix B.1). For all participants except the control group, there were additional, specific questions concerning the support tool (see appendix B.2). Meanwhile, the experimenters met and discussed what video sequences to use for the MWL phase: the protocol and a brief study of the video tape was used to select events. The sequences were selected as described in section 5.5.2 below.

After having filled out the questionnaire, the MWL rating phase began. The participants were seated in front of the TV monitor in the room (see figure 5.2) and shown two to three short (1–2 minutes) video clips from the work on the design. In the cases where the sequence involved a comment, the tape was started approximately one minute before the delivery, and run until the participant indicated that they recognized the comment and recalled the situation. In the case of a non-comment sequence, the tape was run until the participant recalled the situation.

After each video clip, the participants' mental workload was measured using RTLX (see appendix C.1), which required the participants to fill out a form. In case of a comment sequence, they also completed a short questionnaire concerning their opinions about the received comment (see appendix B.3). Each sequence was ended with a short, semi-structured interview where additional comments from the participants were collected.

After the MWL rating task, each participant was interviewed by the experimenters. The interview questions were individually composed, during the session, and concerned the design of the environment and the participant's perception of it. The aim was to collect qualitative data based on problems or "odd" situations observed by any of the experimenters.

Before leaving the lab room, the participants were reminded of the second session, but nothing was said about the nature of that task.

5.5.2 Selection of video sequences

For each participant, a set of two to three video sequences were selected from the recording of the design task. The selection of the sequences were made immediately after the design task, at the time when the participant filled the background data and general rating questionnaire. The two experimenters met and discussed situations to select by studying fragments of the recording. They also used notes, made during the design task, to guide the selection.

For the S groups, 1–2 of the sequences were taken from situations where a comment had been delivered. The rest of their sequences, and all the sequences for the control group, covered situations without comments. Each sequence lasted for 1–2 minutes.

The situations were selected using the following guidelines:

- Each situation should be approximately 60 seconds and cover either a comment delivery/response, or some sort of “design activity”, e.g. creation of a specific object, layout of a sub-part of the interface, alignment of a set of objects, etc.
- For each S participant, different comments should be covered.
- For each C participant, 2–3 segments of the recording should be selected at random (1–2 for the S group), with the following constraint:
- Each situation should be recognizable by looking at the video for 1–2 minutes. Hence, the situations should be either a) design “activities” that are that short, or b) the beginning or ending of a longer “design activity”.

Each sequence was presented for the participant by starting the video 20–30 seconds before the actual situation, and run until the end of the situation, plus a few seconds. Figure 5.5 depicts the sequence selection and replay heuristics.

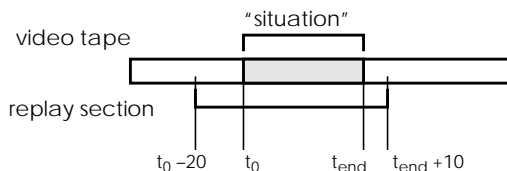


Figure 5.5 Selection of video sequences for measuring specific ratings and perceived MWL

5.5.3 The evaluation task

After 7–10 days, each participant returned to attend the second session, to perform the evaluation task, which lasted between 15 and 30 minutes. This session took place in the same lab as the first session, at another desk in the room, as shown in figure 5.2. After having entered the room, the participant was seated at the desk, and given (approx. 3 minutes) instructions. Two experimenters were present at each session, seated behind the participant. All participants worked under the same conditions.

The participant was told to look at and comment upon a set of three different proposals, presented as images on paper, to the design task presented at the previous session (see appendix D.2). The three proposals were described neutrally; nothing was said about the fact that they contained design flaws. Each participant was allowed to deliver the comments any way they wanted to, written or spoken. However, they were encouraged to describe each comment using gestures and voice, in order to record it using the video camera mounted above the desk.

After being told about the task, the participant was given a pencil, the three proposals and the written specification from the design session. Each participant was told to work for as long as they wanted (we had told them that this session was to last for approximately 30 minutes).

Throughout the session, the participant was allowed to ask questions concerning the evaluation task as well as the written specification. However, the experimenters did not allow for discussion of design solutions, work strategies or evaluation knowledge.

The evaluation was finished when either a) the participant chose to finish, or b) more than 25 minutes had elapsed and the participant did not produce any comments voluntarily.

After finishing the evaluation phase, the participant turned towards the experimenters, and the debriefing interview began. After the introductory questions about whether or not the participant had thought about or discussed the design session during the past week, the actual debriefing took place. An introductory set of questions aimed at finding out if the participant had realized the real nature of the experiment, and a following discussion exposed the setup. After this, each participant was asked for their permission to let us use the data.

A few more questions were used to collect qualitative data. Those questions concerned the general perception of the experiment, the perceived usefulness of the commenting system and the participant's current

(renewed) opinion about the design session. A short period of time was allowed for free comments by the participants. After this, the participant received the reward (a movie ticket).

5.5.4 Expert evaluation

After the completion of all the participation sessions, the expert evaluation data was collected. For this phase, three researchers were asked to evaluate the same user interface proposals that the participants had evaluated earlier. The three researchers, actively solicited, have several years of practice in work with user interface design, in professional settings as well as in academia. All of them knew about the CODEK project, and the Wizard-of-Oz approach, but none of them had seen any material related to the design or evaluation tasks beforehand. All of them had seen but not used the UIB.

The material delivered to each expert included the written specification of the hotel system and a written task description, equivalent to the one told to the participants during the evaluation session (see appendix D.1). In addition, the description included an overview of the capabilities of the UIB used in the design of the proposals. This was to compensate for the lack of practice in comparison to the participants. The experts were also told to include a comment *only* if they considered it to improve the usability of the resulting application.

The experts evaluated the proposals independently at separate occasions, without discussing their work. They performed the task alone, allowed to work for at most 30 minutes. After having completed their tasks, the experts returned the material, including their contributions.

5.6 Collecting, coding and analysing data

Several different techniques and instruments were used to collect and describe data during the experiment. The following summary is given in order to make it easier for the reader to interpret and analyse our findings.

5.6.1 Questionnaires

The basic instrument for collecting data was a set of questionnaires. Those were used for two reasons: they provided background data about the participants and were the main source of data for the usefulness and appropriateness questions. The answers were coded, stored and studied in StatView and Excel. Analysis of variance (ANOVA), linear regression

analysis and analysis of correlation (as defined in Excel) were used to interpret the data.

5.6.2 Design session video recording

The video recordings from the design sessions were used mainly for measuring the perceived MWL, i.e., sequences were played back to the participants immediately after the design session. After the experiments, they were used by the experimenters for timing and clarification purposes, when studying interaction log data and MWL ratings.

5.6.3 Interviews and free notes

After both sessions, semi-structured interviews took place, the aim being to collect comments and additional preferential data from the participants. All conversation was recorded on video, and additional free-form notes taken by the experimenters. The information collected during the interviews was not formally analysed, but nonetheless studied during the analysis of the quantitative data. Both experimenters also took notes during the design session, trying to describe the participants' activities.

5.6.4 Interaction log analysis

During the design sessions, all actions performed in the system were recorded in the interaction log files (see section 4.3). This information was to be used for investigating the possibility of correlating perceived MWL with user interaction. A positive result would indicate the possibility of adjusting the commenting strategy in accordance with the current situation.

The data in the interaction logs was coded and used to plot interaction diagrams (ID), as the ones depicted in appendix E. The ID shows the events that took place during the entire design session, from the point where the participant began to work to the last action performed. Each ID contains plots of five different kinds of data:

Single event plot

The start and duration of each single event type (*create*, *select*, *copy*, *edit*). Note that there are two types of events: actions with duration (e.g., *move*, *set property*, *align*) and actions without duration (e.g., *create group*, *open attribute editor*).

Total event plot (event density)

The sum of all the single events in a single density plot.

Video replay sequence plot

The start and duration of each video sequence replayed for the participant. Each bar is labelled with the perceived MWL value and the start time.

Floating sum and average graphs

Two of the compound values calculated for correlating interaction and perceived MWL.

Comment event plot

The delivery time and duration of each type of comment from the Wizard. Each comment bar contains markers for the “select comment” events.

Three different compound measures were calculated from the set of component values: *transient sum*, *floating sum* and *floating average*. For these calculations, the log was separated into time slots of 30 seconds. Transient sum is the total number of initiated or ongoing actions at the current time. Some actions, like *create* and *align*, have duration, i.e. start and stop times. Actions like *create group* and *open attribute editor* have no duration.

The floating values were calculated from the total sum (average) of events for the last 3 minute period, using a floating 3 minute “window”. The 30 second interval was chosen arbitrarily, but with the rationale that the interval should be long enough to contain interaction data, yet small enough to be useful for creating rapid commenting strategy adjustment. The 3 minute interval was considered to be long enough to contain a “design situation” in detail design work, and yet small enough for strategy adjustment.

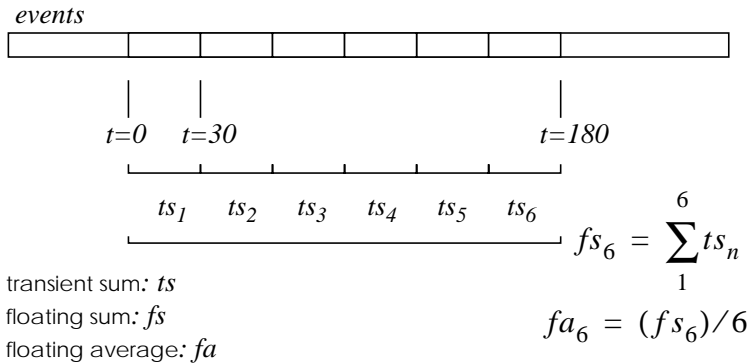


Figure 5.6 Calculating floating and transient interaction values

The compound values were used to correlate interaction log information with perceived MWL, situation assessment, etc.

5.6.5 The evaluation session and expert evaluation

The video recordings from the evaluation sessions were used to analyse the participant's evaluation behaviour and possibly changes in behaviour from using a commenting tool. All the participant's actions and comments were recorded on video, and additional notes concerning comment behaviour taken by the experimenters. All the interface proposals presented were stored, as some participants marked comments on those.

The video, the proposals and the notes were used to classify the participant's comments and commenting behaviour. The session and each utterance from the participant was described using the following dimensions:

Time

At what time (counting from the start of the evaluation task) was the utterance made?

Objects

What object(s) is/are involved? Each utterance refers to: interface number and object identity (all objects had been labelled before the analysis).

Level/type

What kind of utterance is it? Does it refer to lexical, syntactic or domain-specific issues, or information from the specification?

Origin

How "valid" is this utterance? Does it refer to a planted flaw, a potential flaw or is it erroneous? The coding of the origin of utterances is explained in table 5.1.

Mood

In what way is the utterance made? Is it imperative, declarative, a suggestion or a question?

Argumentation

Does the participant present any kind of rationale behind the utterance? If she does, what does she refer to: the specification, the domain, style guides, guidelines or information from the commenting tool?

Origin	Definition
Planted	the comment clearly refers to a flaw that has been planted by the experimenters
Erroneous	the comment refers to something that is judged to be definitely allowed or encouraged in the guidelines used during the design session
New	the comment refers to a flaw that is not planted by the experimenters, but judged to be a flaw since it violates either a) the guidelines used during the design session, b) the source documents used for that subset of guidelines, or c) comments from the expert evaluators
Possible	the comment refers to something that the experimenters cannot clearly define as a violation of the guideline set from the design sessions, but possibly as a violation of other commonly used guidelines or style guides

Table 5.1 Coding the origin (i.e. validity) of participant utterances

After this classification, each utterance was matched against the guideline database used during the design session. Utterances that were considered equal in content to a guideline were labelled with the corresponding guideline number (1–33). Other utterances were used to label new comments (34–45), and grouped if equal.

The expert evaluation was used to provide a way of judging the usefulness of the participants’ utterances, and thereby provide some sort of competence measure of their evaluation. The experts provided their input in written form: interface proposals with annotations and additional free-form comments. Their evaluation data was coded, classified and used to generate new comment labels (46–61) in the same way as the participants’ evaluation data.

From the experts’ data, a new dimension was added to the previous ones in order to provide information concerning knowledge transfer effects: “useful comment”. We worked with two different usefulness definitions: a participant utterance was rated useful if the corresponding comment had been given by 1) one expert, and 2) two or more experts.

Chapter 6

Results

Before using the results to draw conclusions, we present background information and data about external factors of importance for the interpretation of the data.

6.1 Wizard performance

In order to make it possible to interpret and review the results, we first present some data related to the actual performance of the Wizard. Three factors, to a large degree depending on the Wizard, have direct impact on the rest of the data: the number of comments delivered, the delivery delays and the types of comments delivered.

6.1.1 Comment volume and delivery distribution

The number of comments and the delivery speed may be important for the interpretation of differences between the active and passive mode. Since the commenting tool is simulated by a human, raw speed and computational power is most probably lower than that of a full implementation. If the goal of this kind of tool is to produce as many comments as possible, the human Wizard is probably inferior to the implementation. Table 6.1 shows the distribution of the total number of comments delivered for each of the experimental conditions. The number of comments given, distributed by experimental conditions for the mode variable only (active, passive) is shown in figure 6.1³.

³ In this chapter, we present overviews of results using standard bar graphs and graphs showing distribution. The latter graphs consist of a bar showing the min/max and a single line showing (mean +/- standard deviation).

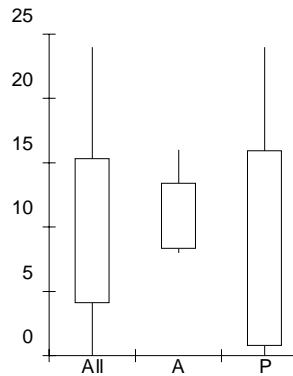


Figure 6.1 Number of comments delivered in the active and passive condition

	Total	Mean
active	76	11.3
passive	58	8.3
declarative	48	6.9
imperative	78	13

Table 6.1 Wizard performance in terms of comments delivered per experimental condition.

It is clear that the passive condition yields greater variability, ranging from 0 comments to 24 for the current design tasks. This kind of behaviour, a great variability in the number of delivered comments, would probably be the result also when using a real passive tool. The range of variability for participants in the active condition is much smaller: from 9 to 16 comments. They also average more comments than the passive groups. This shows that the Wizard is capable of working rather consistently in that there is a small variability in the number of comments generated when the participants do not ask for them.

6.1.2 Delivery speed

An important factor in the passive commenting process is the delivery delay—how long is the user willing to wait? Due to difficulties for the Wizard to interpret the request and create relevant comments, in combination with delays in the data communication, some delivery delays became rather long. In the active case, delivery delay becomes a complex issue,

hard to define since there is no well defined request moment. We have chosen not to study that factor here. Figure 6.2 shows the distribution of delay times in passive comment delivery, elicited from the log files.

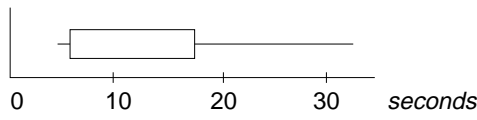


Figure 6.2 Delay time for comment delivery for the passive condition.

6.1.3 Types of comments delivered

We study the perceived usefulness of comments, hence the nature of the comments is also interesting, in addition to performance data. Figure 6.3 shows how many times each individual comment was delivered. The Wizard did not apply any of the other fifteen comments (4, 5, 8, 11, 14, 15, 17, 21, 24, 25, 27–31) during the design sessions. This may indicate that the comment set was perhaps not optimal for the task (see section 8.2.3).

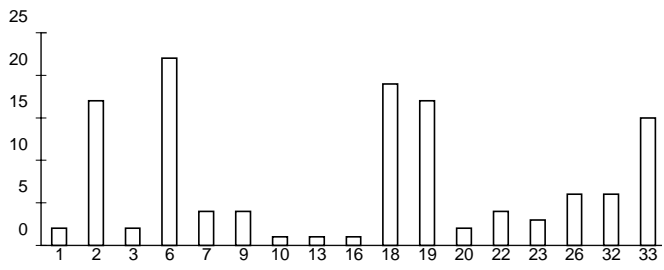


Figure 6.3 Number of times each comment type was delivered.

6.1.4 Video sequences, MWL measures and questionnaires

A total of 45 video sequences were used for collecting information about the participant’s perception of specific situations and their perceived MWL. Each person viewed two to three sequences. This resulted in 37 data points for the groups that used the simulated commenting tool, and a total of 26 data points where a comment had been delivered (see table 6.1).

6.2 Participant perception of task

Besides the Wizard performance, the task and participant perception of the task are important for the interpretation of the results. A set of general

Total number of sequences	45
Number of sequences for the S groups	37
Number of sequences with comment situations	26

Table 6.1 Number of video sequences, i.e., data points for specific situations. questions concerning the environment, the participant’s effort, the participant’s perception of the task, etc. were used to collect information about this. All participants were asked to rate the following:

- Task complexity.
- Satisfaction with their own solution.
- Uncertainty about the task.
- Stress perceived during the task.
- Effort during task.

For each question, an 8-point Likert-type scale was used to collect data. In the presentation below, each scale is referred to in a uniform way: one is the lowest value and eight the highest. Analysis of variance of average group ratings were used to detect differences between different experimental conditions.

The task complexity was rated below medium (3.7) and all participants were satisfied with their solution (5.9). The reported perceived uncertainty about the task was low (2.9). For these three measures, no significant differences were found between the groups.

Participants reported putting in medium effort (4.5) while solving the task, with no significant effects between groups. Perceived stress was rated as rather low (3.2), with a significant effect from different moods. Participants in the imperative group reported a higher amount of perceived stress than the declarative group during the task (3.2 vs. 1.9, $F(1,11)=3.9, p<0.1$).

6.3 Usefulness of a commenting system

The usefulness of a commenting system was investigated by asking the participants to rate the system and the single comments received. Two types of rating were used: 1) immediately after the design task the participant rated the *general* experience, and 2) after each video sequence replay the participants were asked to rate the system as perceived in that *specific* situation. The overall questions addressed the usefulness of the comments, the perceived supportiveness/disturbance of the system and the participant’s opinion on using such a system in the future. In each of the specific

ratings, the participant answered questions similar to the general questions. An additional question addressed whether the tool had provided the participant with information that was new to them.

Commenting tool—general and specific usefulness and helpfulness

Questions 3:1, C:1, 3:7, C:6, 3:5

The participants working under the commenting condition rated the comments to be useful in general (6.6) as well as on average in the specific situations (5.4). They found the comments (i.e. the commenting system) to be helpful in general (5.7), and in specific situations (5.0). In general, participants answered that they would like to use a similar tool for similar tasks in the future (7.1). The ratings for these three questions are consistently above the mean of the scale (i.e., greater than 4.5).

Total environment—general supportiveness and disturbance

Questions 2:8, 2:7

In comparison with the control group, the participants using the commenting system rated the environment as being slightly more supportive (5.8 vs 4.7, n.s.), and at the same time somewhat more disturbing (3.7 vs. 1.7, n.s.). These ratings cover different amounts of the systems for the different groups S and C. For the control group the total environment equals to the design environment only, and for the participants in the S group the total environment includes the design tool and the commenting tool.

6.4 Effects of different commenting strategies

In order to investigate the effects from using different commenting strategies, the participants in the groups using the simulated system were asked to rate the commenting system and the comments delivered. Two sets of questions prompted for the participant's perception of the system, as related to the current mode and mood. The questions used to investigate the perception of different modes concerned the degree of disturbance perceived, the supportiveness of the system and the speed of comment delivery. A total of eight measurements concerning comment delivery were available from the experiment.

6.4.1 Mode

For the mode dimension, the relevant questions addressed supportiveness and disturbance, comment system speed, plus the participant's ability to

find an object pointed out in a comment. We found five significant effects from different mode settings.

General supportiveness and disturbance Questions 2:8, 2:7

The environment was rated significantly more supportive by the active group than the passive one (6.6 vs. 5.0, $F(1,11)=6.4$, $p<0.05$). The overall average measure was 5.6. Different modes produced no significant differences for the ratings of disturbance (3.7 vs 3.6, n.s.).

General and specific speed Questions 3:2, C:5

Different modes produced a difference for the ratings of delivery speed. In the general case, participants in the active group rated the system as somewhat faster than the passive group (4 vs. 3.2, n.s.). For the specific measures, the difference was significant (4.8 vs. 2.8, $F(1,24)=10.7$, $p<0.01$). The average rating of the delivery speed was 3.7 in general, and 4.0 on average for the specific measures. The delivery speed ratings also showed a significant interaction effect ($p<0.05$), as depicted in figure 6.4. Participants using the declarative tool rated the system as being faster if it was active. The group using the imperative tool rated the system as being faster in the passive mode.

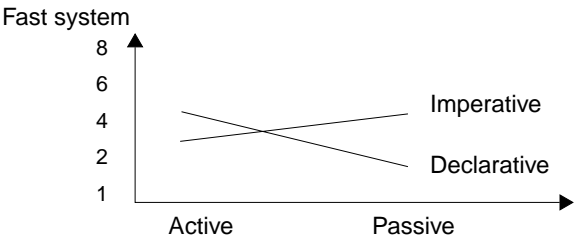


Figure 6.4 Interaction effect (between mood and mode) for the participants ratings of the speed of the commenting system.

General and specific ability to find object Questions 3:8, C:7

The question concerning the participant’s ability to find the object referred to in a comment was applicable in 25 cases. A significant interaction effect ($p<0.05$) was found for this question, as depicted in figure 6.4. In the active mode, participants reported objects to be easier to find if the system was declarative. In the passive mode, they rated objects to be easier to find in the imperative mood. In the specific case, the active group reported that it was easier to find the object than did the passive one (6.9 vs. 3.9,

$F(1,23)=8.8, p<0.01$). The average rating was 5.3 in general and 5.7 on average in the specific situations.

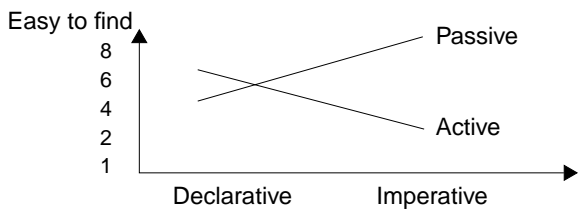


Figure 6.5 Interaction effect (between mood and mode) for the participants ability to find referred objects

In addition to the quantitative data, most participants commented upon their preferences concerning the mode dimension during the interviews, after both sessions. From those interviews, we observe that participants receiving active comments claim to prefer a passive system and vice versa. Participants from the control group want to be able to choose mode freely, for the whole session as well as runtime adjustment.

6.4.2 Mood

For the mood dimension, the relevant questions addressed the comprehensibility of the comment, the perception of the competence of the tool and the participants ability to find objects referred to in a comment. We found three significant effects from different mood settings.

General and specific comprehensibility and system competence

Questions 3:3, C:3, 3:4

Participants using the declarative system rated comments as more difficult to understand than those using the imperative system in general (6.3 vs 2.8, $F(1,10)=14.04, p<0.05$) and on average in the specific situations (5.7 vs. 2.6), $F(1,24)=10.1, p<0.01$). The overall average rating was 4.6 in the general case and the specific average was 4.0. Those using the imperative system rated the system competency lower than the declarative group (4.7 vs 5.8, $F(1,10)=4.0, p<0.10$).

In addition to the data collected using the questionnaire, it was observed by the experimenters that some comments were not noticed at all by the participants. This was verified by the participants themselves during the video sequence replay. Hence, all the delivered comments were

classified according to whether the participant noticed them or not (see section 7.3).

6.5 Transfer of knowledge/behaviour

Participants' previous knowledge

Questions C:4

Before being able to evaluate knowledge transfer, the participants' familiarity with the (new) information must be investigated. Participants rated the information to be new or partly new to them (3.1), with no significant differences between groups.

Total evaluation time and commenting activity

The participants spent on average 11.5 minutes evaluating the proposals. Each proposal was given on average 4 comments. There were no significant differences between the groups.

Participants delivered on average 17 comments during the evaluation task. Each participant delivered on average 5 different kinds of comments. The control group delivered more comments than the other groups (24.3 vs. 15.0, $F(1,14)=2.78$, $p<0.12$). No other significant differences in volume or type were observed between groups. Figure 6.6 shows a summary of the comment volume and distribution for the different groups.

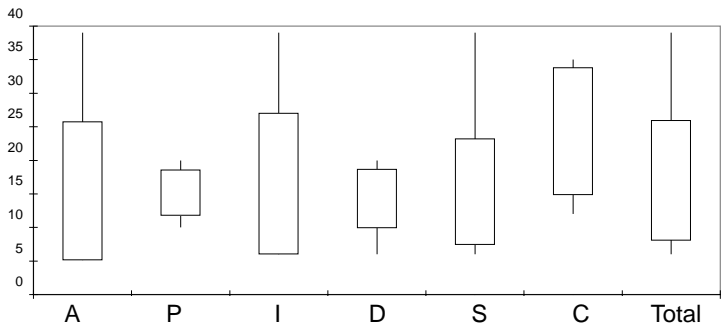


Figure 6.6 Comment volume and distribution for the different groups during the evaluation task (all comments).

Knowledge transfer

To investigate the potential knowledge transfer, we started by looking at the ratio useful/not useful comments, as defined by the expert evaluation (see section 5.5.4). On average 73% (60%)⁴ of each participants com-

ments belonged to the useful comment types. In terms of volume, the slight difference between the control group and the other groups remained (18.7 vs. 11.2, n.s.). There were no other significant differences between the groups.

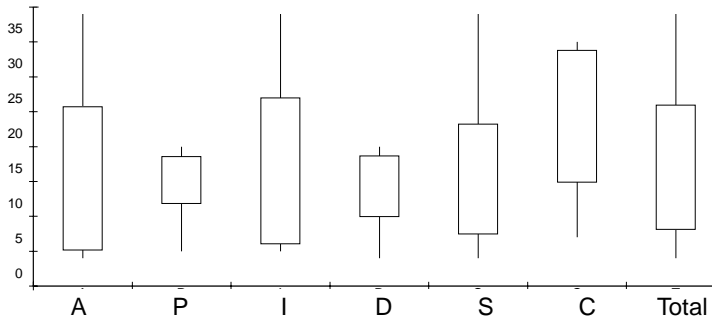


Figure 6.7 Comment volume and distribution for the different groups during the evaluation task (only “useful” comments).

Another way to look at potential knowledge transfer is to investigate the difference in comment types delivered by different groups, i.e., to compare the S groups with the control group. The comment types and volumes of different comments for each group is presented in figure 6.8. Note that the number of data points is rather small, which makes comparisons on a more detailed level (i.e. for specific comments) hard. There were no significant differences in volume/comment between the S and C groups.

Each comment and utterance was classified in terms of level, origin, and argumentation, as described in section 5.6.5. A comparison between groups for these codings showed few significant differences. The control group used more questions and suggestions compared to the other groups (7 vs. 2.8, $F(1, 14)=6.4$, $p<0.05$).

As mentioned in section 5.6.5, the design proposals were likely to contain design flaws not covered by the knowledge-base guidelines. Several participants identified “new” problems and potential design flaws, as did the three experts. The final classification of comments contained 62 design comments, i.e., twice the amount of items of the original comment set.

⁴ For the comparisons and graphs presented here, a useful comment is defined as one which *one* expert has rated useful. However, for some measures we also present the corresponding result for the harder definition (requiring *two* experts to agree).

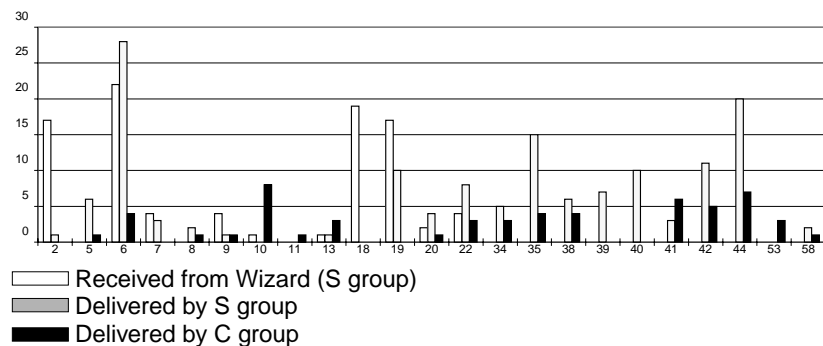


Figure 6.8 Number of comments (for each comment type) delivered by the different groups.

6.6 Perceived MWL

A total of 45 measurements of perceived MWL were available after the study. The total average MWL rating was 41.7 (on a scale 0–100). A comparison of the control group (C) and the rest of the groups (S) shows that the participants using the simulated commenting environment reported a higher perceived MWL (45.4 vs. 24.9, $F(1, 43)=19.1$, $p<0.01$). There was no significant difference between situations with and those without comment delivery.

The mood condition produced a significant difference in MWL ratings; participants in the declarative mood reported a higher value than those in the imperative group (52.0 vs. 37.6, $F(1,35)=16.7$, $p<0.01$). There were no significant differences for different modes. Figure 6.9 shows a summary of the MWL ratings for the different groups.

In order to interpret the validity and importance of the MWL we compared the MWL with some of the related questions in the questionnaire: perceived disturbance and support. This comparison was applicable in 24 situations (see section 6.1.4). MWL was found to correlate rather well with both perceived disturbance and supportiveness: positively with the former ($\rho=0.60$) and negatively with the latter ($\rho=-0.53$).

6.7 Interpretations of the interaction logs

The interaction logs were used primarily to guide the interpretation of MWL data and follow-up studies of video sequences. We found no strong correlation with MWL measures or questions for specific situations for

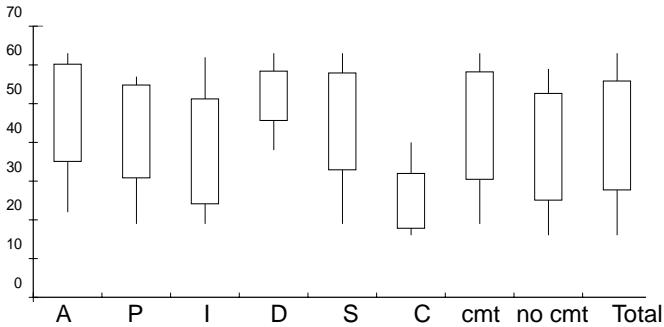


Figure 6.9 Perceived MWL distribution for the different experimental conditions.

any of the compound measures of interaction from the ID (transient sum, floating sum and floating average).

We note that the mere graphs of the ID:s in some sense provide information about the participant's design behaviour. The single event plot sequences form different patterns for different participant behaviour. Appendix E contains printouts of three ID:s: ID 9, ID 14 and ID 16.

In ID 9 the participant creates a small number of objects, and immediately tries to locate them in their final position, which is seen from the low number of move-actions. The participant immediately adjusts all attributes necessary, and then requests comments. This sequence is repeated throughout the session. Only at the very end does the participant delete some objects.

ID 14 indicates a totally different behaviour. This participant first creates the main part of the objects needed for the interface, making only a very rough position adjustment. The participant then switches between adjusting position and attributes for the objects for more than 15 minutes. This is followed by a phase where creation, adjustment and deletion is interleaved for approximately 15 minutes. The last 20 minutes are spent requesting comments and adjusting the interface accordingly. This ID in some sense shows a session where comments are requested too late.

ID 16 shows a situation where the participant has been working for 20 minutes, creating, moving and adjusting objects to build a large part of the interface. Then the participant decides to attend to the comments delivered (active mode), which stalls all design activity completely for over 4 minutes.

6.8 Participant comments and interview summary

During the two sessions, data was collected using observation and via interviews. This data was not intended to be used in the current study and is therefore not presented in detail here. However, some issues are of interest in the current context.

6.8.1 Comments after the design session

After the design session, each participant was asked if she had any particular comments regarding the system, and whether she wanted things “differently”. More than half of the participants using the simulated system suggested that some sort of examples should be included in the commenting system, or as a complement (e.g., via a hypertext system). Almost all the S group participants also indicated that they would have preferred (to test) a system with the *other* mode (i.e., participants using the active system would like to test a passive one). A few of them proposed the idea that the system should be either adaptable (the user may choose mode) or adaptive (the system adjusts the mode automatically).

6.8.2 Interview and “deprogramming” after the experiment

After the evaluation session, the participants were “de-programmed” and the Wizard-of-Oz setup described. After having debriefed the participants, telling them about the actual setup, everyone was asked if they allowed us to use the material collected during their particular sessions. As is the case in most reported Wizard-of-Oz studies, everyone agreed. In most cases, they also commented upon the setup in a positive manner: Several participants found the approach itself interesting and was happy to discuss it with us afterwards:

“I did never realize that I was ‘talking’ to a human—that was fun!”

“That was an interesting experience—I have only read about this kind of experiment before.”

However, we also experienced some problems, possibly due to the fact that we did not debrief our participants until after the second sessions.

One participant complained

One participant disagreed with our setup after having been debriefed. This person argued that we should have used other methods, instead of the Wizard-of-Oz setup, which s/he perceived as “disturbing”. Together, we discussed the outcome and the situation for a while, immediately after the

second session, and tried different alternative solutions. The participant allowed us to use the collected data without complaints.

Chapter 7

Conclusions

We now move on to present a set of conclusions in order to achieve our original goal—to inform future research and to guide the design of future design support tools. Although it may be argued that the sample used for this study is too small, some results are strong, and backed up by several findings.

7.1 Usefulness of commenting tools

The results from the questionnaire indicate that a commenting tool is perceived as useful in the support of user-interface design. The participants rated the commenting tool's usefulness and helpfulness above average. At the same time, the tool is perceived as somewhat disturbing. Seeming contradictory at first, this is not at all problematic if we consider the finding that the comments provided the designers with previously unknown information. Of course, I might appreciate someone telling me things I don't know, but that are useful to me. At the same time I might find this disturbing, if it interrupts me in my current task.

The usefulness confirmation is supported by results from a related study by Löwgren and Laurén (1993). Their results show that this kind of tools are seen as potentially useful by professional designers.

7.2 Appropriateness of comments

Our study showed a clear preference for imperative comments over declarative ones. Although our distinction between declarative and imperative sentences was about linguistic mood only, it is an indication that the dif-

ference between telling someone what to do might be more effective than just informing them about a possible flaw, in this domain.

Informal backup for our results is provided by comments recorded during the interview after the design session. Most of the participants in the declarative group suggested that an imperative approach would be preferable, and often referred to the possibility of presenting/showing positive examples in addition to the comment. They even extended their suggestions from the basic mood dimension chosen here, to the possibility of using different depths of content in the presentation.

As seen in section 6.4.2, the imperative tool is rated somewhat less competent than the declarative one. This will of course affect the designer’s perception of the system, and possibly her willingness to use it.

7.3 Perceived MWL and comment detection

Our results indicate a relation between perceived MWL and the user’s ability to detect comments. The distribution of the perceived MWL and their ability to notice each comment is shown in figure 7.1.

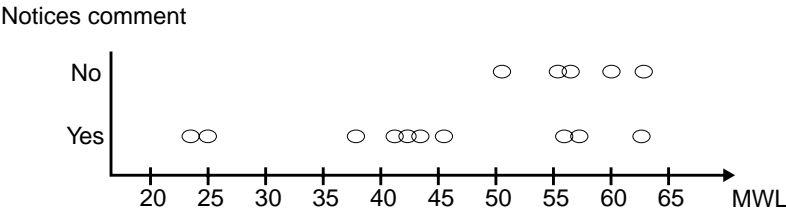


Figure 7.1 Perceived MWL versus ability to notice comments

It is clear that for situations where the delivered comment is noticed by the participant, the perceived MWL is rather evenly distributed. For those events where a participant did *not* notice a comment, the perceived MWL mainly lies in the higher range of the scale ($RTLX > 50$).

In the three cases where a participant noticed a comment despite a higher MWL, the comment and the visual indication concerned an object situated close to the area or object they were currently working on. Those comments were also rated less useful and more disturbing (3.0 compared to the overall average of 5.1) than the others. These results suggest that people do not notice a comment when they are intensely busy with their design activities, unless it falls within the field of visual activity.

The identified relation between perceived MWL and ability to notice comments may be interpreted in at least three ways, each one suggesting a different technical solution.

Timing is unimportant

The exact timing of delivering a comment is not important. When the user's MWL is low enough, she will see it. Comments could be delivered at any time, which means that no special temporal mechanism is needed.

A temporal strategy is needed

It is useless to present a comment when the user's MWL is too high. The comment might even increase the MWL, causing the designer to ignore the comment, or treat it as another disturbing element. A study by Carroll and Aaronson (1988) showed that users were not able to utilize the help (from a simulated active help-system) if it did not appear at the appropriate time. In our system, this means that the commenting tool would need a temporal strategy—an algorithm delaying comments until the user's MWL has lowered. This requires that the system could somehow estimate the user's MWL.

Better cues are needed

There are comments that really should be read as soon as possible by the user, i.e., the attention must be drawn to the comment immediately to ensure that the violation is detected before it is too late (c.f. Lemke and Fischer, 1990). More powerful visual cues and presentation techniques are needed in the tool, e.g. sound signals and colouring of violating objects.

7.4 Knowledge-transfer effects of using a commenting tool

We identified no significant effects in terms of knowledge transfer. Hence, we cannot conclude anything on this matter. Our study resulted in a very small number of data points for analysing transfer; each person delivered on average less than 5 comments per evaluated interface, and less than 5 different kinds of comments. This, in combination with the fact that the Wizard managed to deliver on average only 9 comments per participant, made correlations of delivered/received information difficult.

However, when observing the video recordings and evaluation material, we did find some indications for differences in commenting behaviour

between groups. It seems as if users of a commenting tool behave somewhat more “carefully”, in that they deliver fewer comments, but present those more firmly (e.g., as conclusive remarks and proposals) than people with no experience from using commenting tools. We also noted that those who have received comments delivered more higher-level comments (i.e. task-related) than the control group.

7.5 Correlation between activity measures and subjective situation ratings

The reason for performing explorative studies of potential correlation between interaction data and the designer’s perception of the situations is simple: even a non-intuitive, illogical, non-structured measure would be valuable if it could be used to improve the delivery process, i.e., if a temporal strategy is needed. However, our studies indicated only a vague relationship between any single or compound interaction measure and the participant’s situation ratings or perceived MWL.

The data resulted in a very low correlation between perceived MWL and the participant’s rating of the situation, e.g., comment usefulness, comment comprehensibility or comment disruption. This may be interpreted in several ways. It could mean that any (or both) of the two measures is invalid, i.e., that our data collection was improper. However, by studying the participant’s answers, the video sequences and the interaction logs, we consider the situation ratings to be consistent within as well as between participants and groups. Another interpretation of the non-correlation is that it shows the complexity of perceived MWL measures. An incomprehensible comment, which interrupts the participant and is rated low on the usefulness scale may be rejected quickly and hence result in a low MWL. Similarly, a useful, non-disturbing, clear comment may cause the participant to engage in concentrated reflection and analysis of the situation, which implies a high MWL.

Low activity cannot be used as an indicator of high acceptability of comment delivery. There are too many different interpretations of low activity, e.g. the MWL is too high; the designer works very slowly or inter-leaves intense activity with concentrated reflection; something interrupts the designer who perhaps leaves the workstation. In several ID:s (e.g. ID 16) activity suddenly drops to zero, but this is because the participant brings all her attention to the delivered comments, and then finds them hard to interpret. Additional comments delivered at that moment would

either disturb or be neglected. The crash and restart in ID 9 in some sense illustrates a designer who is interrupted, e.g., by the phone. During the inactive 3 minutes, the designer would not be able to attend to comments.

We explored the possibility to use the raw interaction measures for a strategy guiding algorithm, i.e., to predict the user's rating of a situation based on interaction. Using multiple linear regression and an *ad hoc* process when selecting components, we tried to develop a predicted measure of perceived disturbance. The input data consisted of the raw interaction information from 26 situations (data from different participants) where questionnaires and MWL forms had been used. In the "best" model achieved, the predicted value is based on 9 components (transient number of events for 10 kinds of events), resulting in $R^2=0.66$ and justified $R^2=0.47$. Figure 7.2 shows the correlation between the participant's ratings and the predicted values for the 26 situations. However, the resulting model is obviously very weak and sensitive; a much larger number of data points and a better procedure for selecting event components are required.

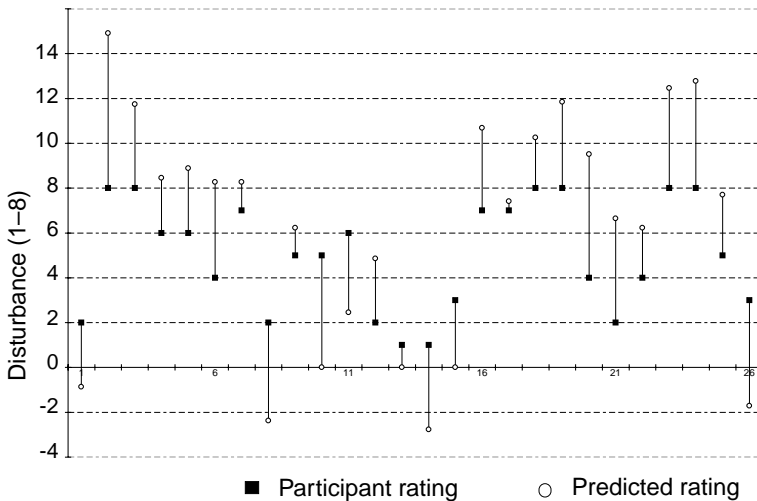


Figure 7.2 Predicting the participants "disturbance" rating using raw interaction data. The x-axis corresponds to the 26 situations (i.e., it is not a continuous time-scale). Shorter staples means better prediction.

The value of this technique must be investigated in depth before conclusions can be drawn. We have not had time to collect new data on which the predictive power can be tested.

7.6 Design recommendations for commenting tools

As a final part of our report of conclusions, we summarize our findings as design recommendations for future work on designing commenting support tools for user interface design. This is part of our overall research goal: to improve today's design support tools. We argue for four different implications of our results.

First, we conclude that from a user perspective, commenting tools are perceived as useful. However, we also identify the need for more evaluative work in the area of commenting design support tools. We have added knowledge about the user's perception of such tools, and hence believe that we motivate studies of process and product effects from using this kind of support tools. That would be the first step toward developing new design support tools.

Secondly, since our study indicates that an imperative mood is preferred, process and product studies should verify the effects of this variable. Our results are not surprising, but should nevertheless help guide future research and development; it is better to try to improve imperative strategies than to explore declarative ones. We believe that this is valid on a more general level than the purely linguistic one, but this is something that needs to be tested; is it the case that designers prefer receiving information or knowledge in an imperative mood, regardless of the modality (text, images, animations, etc.).

Thirdly, we can interpret our results from using different temporal strategies in at least three different ways (see section 7.3). All three are closely related to the use of perceived MWL as a measurement, and relies on the basic assumption that the temporal strategy is central in commenting systems. Although we conducted an exploratory study of the use of interaction data to guide commenting strategies, we can claim to have shown neither the usefulness nor the impossibility to utilise low-level user actions as a control parameter in the commenting tools' temporal mechanisms. Our conclusion is that future studies of temporal strategies are needed, and that research concerning interaction data should be conducted.

Finally, data collected at the informal interviews suggests that examples would be valuable, at least from the designer's perspective. This is supported by comments from participants, observations during the pre-study, and also claims by, e.g., Reiterer (1994) and Harstad (1993). Although our participants only suggested the use of positive examples, we

believe that negative cases would be equally valuable during the design process. The form of the examples is critical, as shown by Tetzlaff and Schwartz (1991). If pictorial examples are used extensively, there is a risk that the designer uses only the pictures and disregards the textual information. Further suggestions and input regarding example-based systems are presented by Blatt and Zacherl (1993).

Chapter 8

Discussion and Future Work

In chapter 3, we outlined a process in which this kind of study was the first step. Before moving on to subsequent activities, the results and conclusions achieved here should be examined. In this chapter, we address this issue from four different perspectives.

First, we want to discuss the applicability of our results, i.e., the usefulness of our conclusions. Some of the issues from the previous chapter are investigated in terms of practical value and generality.

Secondly, there are several methodological issues that we have touched upon in previous chapters, but not treated in depth. This chapter contains a discussion about the validity and reliability of our results. Furthermore, when conducting Wizard-of-Oz experiments, several questions arise concerning ethics and the researchers' attitude towards participants. In order to inform future work using the Wizard-of-Oz methodology, we present our experience and opinions.

Thirdly, there are some issues and questions that we have addressed for exploratory purposes only, and therefore not treated or analysed as formally as the main questions. Those issues, including the use of interaction data for improving commenting systems, are discussed here.

Finally, we want to present our view on the need for further studies and immediate continuations of this experiment. On the one hand, our results and conclusions may be used for guiding the implementation of prototypes to be tested in industrial settings; at the same time, we believe that there is a need for further empirical work.

8.1 Applicability of the results

The main question of our research has concerned the overall usefulness of a commenting support tool in the UI design domain. In addition, we have investigated the effects of different commenting strategies. For the sake of clarity, we summarize the context of our research—the results and conclusions should be interpreted in the following perspective:

Design knowledge/information is valuable

We consider user interface design knowledge or information in the form of, e.g., guidelines, style guides, written specifications etc., to be valuable and important, for reasons stated in chapter 2.

There is an information problem in the design situation

All the information available to the designer is likely to cause some problems, as described in chapter 2. Many different ways to remedy this problem have been proposed—in our research we have focused on evaluative design support tools.

This study is a first step in the process of solving the problem

We have used simulation techniques, a controlled environment, a limited task and not-yet professional designers as participants. As described in chapter 3, we believe that this is a first, necessary step, when addressing the questions of 1) what problems and needs the designer really have, and 2) what techniques and tools could help solve those problems.

The design situation we have studied is characterised by...

...a single designer, solving a single task during a limited amount of time, working alone, using a simple integrated design environment.

We want to address two types of important questions related to the interpretation of the results of our research. First, applicability and generalizability should be discussed: What can be concluded from our results, and what are the implications? The second type of question is related to the practical implications and outcome: What is the value of our research; is it useful?

8.1.1 What can be concluded from our results?

The main results and conclusions from this study concern perceived usefulness and effects of a set of different commenting strategies. Hence, we are able to discuss how a user of a commenting tool will react, and how the

reaction changes according to modified mood or mode. The conclusions presented in the previous chapter are applicable in a rather limited situation, but nevertheless show that commenting tools are perceived as useful. This suggests that such tools are likely to be used, and hence will have impact on the designer's work. What kind of impact this will result in is beyond the scope of this study.

Although this study has focused on information related to detail user interface design, we believe that our results concerning commenting tools as support are applicable in other contexts too. There are other situations where a designer/developer needs to access information that guides and instruct current work, and where the amount of information is vast. The comments delivered need not necessarily be about screen layout and terminology consistency. The effects observed here are likely to be found in situations with similarities to the detailed design task, i.e., visual construction and layout tasks with the characteristics of a design situation (see section 2.1). Nakakoji et al. (1995) have demonstrated the use of a commenting tool as support in an architectural application (for kitchen floor-plan design), and Harstad (1993) uses commenting techniques in a visual design system for voice dialog applications.

A very practical, yet research-oriented, outcome of our study is the experience of conducting a study of user-interface design and support tools. We have gained knowledge about the problems related to the practical parts of interface design, and about the problems that emerge when you try to study the use of a software support tool. Now we know much more about 1) how to plan and develop a study about design support systems and 2) what questions and issues to focus on in our future work.

Limitations of our results

Our results do not fully answer the more general question: *What is the contribution of a commenting system as design support?* We can only conclude that it is likely that a user of a commenting system perceives it as useful in a similar design situation. As mentioned above, questions concerning performance and product "quality" remain to be answered.

What if the system is "wrong", i.e., the comment delivered refers to something that the designer either a) doesn't consider to be a flaw, or b) recognizes as a flaw but still want to use, or c) actually has triggered erroneously. Whatever the reason is, this situation is likely to result in some sort of breakdown. During this study we observed a few situations where some kind of breakdowns occurred. In a couple of cases, the participant

disagreed with the system; some participants chose to delete the comments immediately and continue to work; in a few cases, the participant stopped working for almost minutes and just tried to figure out what to do. Although interesting, we did not have the time to study these matters more closely in this study.

A related problem, concerning the adaption of strategies, was mentioned in the previous chapter—it may be too complicated to choose and adapt the commenting systems behaviour in comparison to the effects achieved. In a Wizard-of-Oz study of advice-giving software, in the domain of graphical statistics applications, Hill (1993) conclude that “it is always possible to code individual strategies, but that there might be too many of them to make advice-giving systems feasible to develop”.

On a more general level, our results do not provide information about the general usability of a commenting tool, e.g., performance issues, impact on the design process, suitability in a project where several designers collaborate on a single task, etc. Neither can we say anything about the usefulness or usability in comparison with other support approaches, e.g., written documents, human-human collaboration, constraint techniques or hypertext guidelines.

8.1.2 Are we just addressing the simple part of the problem?

It is possible that someone gets the impression that we are studying the “simple” parts of the problem of designing usable software: instead of evaluating systems that can only improve surface aspects of user interfaces, resources should be directed towards improvement of the systems development process. However, we argue that studies like this are equally necessary, and that the aim of commenting systems need not at all be that narrow.

First, we consider the use of previous design information to be something that occurs in early as well as late phases of a software design project. Hence, the information may include all kinds of guidelines; not just surface layout issues. We consider the use of written guidelines to be a support tool and a way of transferring knowledge between projects. We also recognize the problems inherent in today’s guideline documents.

Secondly, commenting systems are just one approach proposed for solving problems related to information complexity and overload. Our choice to address this particular kind of tools is based on the lack of evaluative work in the area; in order to guide the development of future support tools, the advantages and disadvantages of commenting systems

should be investigated. This is also related to the fact that the use of certain guidelines and style guides is important and often required in today's software design projects.

Thirdly, neither guidelines nor commenting systems *eliminate* the need for competent designers and skill. As with any other knowledge provider (human or artificial), the receiver must be capable of interpreting and judging the value of the output. In one way, a commenting system is like a grammar or spelling checker in a word processor; such tools do not generate "good" sentences, but provide the possibility to "correct" the writer's ones (and the writer may choose to ignore the corrections). Even a rather simple tool *might* help the writer/designer eliminate flaws early during a project, thereby reducing the resources needed for evaluation and correction later on (but not eliminating them).

Then there is the question about the actual value of written guidelines and the kind of information they convey. It is sometimes the case that such tools are used very late in the design work, more as a technique for fixing already implemented problems than as a support tool for eliminating the problems early. However, as we argue in chapter 2, the design knowledge need not be only about layout and widget selection, but often consist of requirements and guidelines concerning the whole system design.

In the CODEK project we have not included all the knowledge available/needed during a design project, which however should not be interpreted as if we consider the knowledge we have used as the only one needed or sufficient. We are mainly interested in the general effects of commenting as design support.

8.1.3 What about other techniques and components?

As indicated in chapter 2 there are many other ways to approach user interface design support: constraint mechanisms, hypertext systems, automated design, etc. The commenting component offers just one of many services that would be required of an integrated design environment. There are several examples of research projects where other techniques have been tested, but to the best of our knowledge, there are no studies where several different techniques have been compared or reasoned about. Recent research has addressed questions concerning the more complex issues of systems development, e.g. collaboration and experience transfer.

Stone et al. (1993) formulate a number of requirements for intelligent support in HCI design, mentioning e.g. domain orientation and flexibil-

ity—the possibility to work top-down as well as bottom-up. Their requirements are based on similar grounds as presented here: the iterative nature of the design process and the information complexity/overflow. Fischer et al. (1995) tries to support long-term collaboration between designers. Henninger et al. (1995) describes an related approach, which aims at building an organizational memory in which lexical and syntactical user-interface design knowledge is accumulated and enhanced over time.

8.1.4 Integrating support for design and evaluation in the designer's environment

Winograd (1995) argues that new kinds of environments are needed for user-oriented software design; he emphasizes the importance of a responsive prototyping medium—an environment that makes it possible to “communicate” (create and receive feedback) with the artifact being designed. An evaluative component, such as a commenting tool, might be seen as a step in that direction (see section 2.5.2). However, what does it mean to integrate tools for creation and tools for evaluation in the designer's environment?

An important question when dealing with any kind of support technique for design concerns the designer's creativity: Will the technique impose too large limits and constraints on the designer? In what kind of situations will such a technique be more/less useful? In the case of a commenting tool, it may be the case that a lengthy “dialogue” concerning a small set of detail design flaws and comments causes the designer to limit her perspectives, focusing on the improvement of the current part of the design only. Eventually, this may lead to a kind of design fixation; the early decisions that lead to this detail design are not evaluated. Hence, the “real” problems are not investigated or remedied.

Hartson and Boehm-Davis (1993) discuss the support for design knowledge and conclude that “an important research goal in the area of user interface development tools is to discover useful, unobtrusive, structured, and organized ways to integrate the use of principles, guidelines, standards, style guides, and designer rules into the tools without stifling creativity”.

For some systems development projects, and for some parts of the design work, it might be the case that a more “controlled”, engineering-type process is wanted. However, in general we believe that the designer's

creativity and responsibility for the design decisions are crucial for a successful usability-oriented process.

8.2 Methodological issues

In order to make it possible to investigate the value of our conclusions, we present some of the arguments, rationale and actions behind the methodological decisions we have made. In retrospect, there are several parts of our study that, from a methodological point of view, maybe should have been performed in different ways.

First of all, there are several factors that we did not address during this study, but which we nevertheless consider important when evaluating a design support tool. First, *product* impact and the resulting user interface quality and completeness is of course a critical factor. Even if the designer perceives the tool in itself as useful, this does not necessarily mean (by the measures used here) that the resulting piece of software is useful (or usable). Although complex to measure and use for comparison, we believe that it would be possible to use, e.g., heuristic evaluation for generating such data.

Secondly, the *process* impact is equally important. A support tool will most probably not be used if it does not allow a decent level of productivity in an industrial setting. Another process-related issue probably rated critical is the impact on transfer of competence and experience to succeeding projects. This dimension is perhaps even more complex than the previous one; at least from the point of measurement technique selection.

The reasons for not addressing any of these issues during this work are partly the limited resources, but primarily related to our view on the approach to studying software support tools (see chapter 3). We now have a lot of information that can help us develop a robust study of professional designers in a professional context. Via the observations and video studies, we have gained knowledge not only about the design process but also on the requirements on the research instruments needed.

We now turn to the issues of validity, reliability and potential weaknesses in our experiment design and methodology.

8.2.1 Validity and reliability in Wizard-of-Oz studies

Of course, our results suffer from the small number of participants in the experiment. However, the real issues of validity or reliability lie not in the

limited set of data, but rather in the experimental setup—the simulation and the artificial environment and task.

A crucial validity question is “what do we really measure, using the Wizard-of-Oz approach?” There are several components involved in this study, closely related to each other, all of which may or may not contribute to the data we collect:

- the commenting component, as perceived by user.
- the user interface builder environment, as perceived by user.
- the Wizard’s behaviour and competence.

How do we make sure that the results concerning the user’s perception of the commenting component is not affected by the other two? Furthermore, how do we take into account that it is the human simulating a computerized agent that is responsible for the system’s behaviour?

Measuring the perceived usefulness of the system

The experiment and the software environment used were designed in order to assure that the data we collect concerning usefulness and appropriateness primarily mirrors the participants’ perception of the commenting component. First, by including a control group, using only the user interface builder, we got a base line measure that could be used to make comparisons. Secondly, we asked the participants to rate the user interface builder separately, and also designed the builder with this issue in mind. This made it possible to extract the information specific to the commenting component from the participants’ ratings. We tested it in the pilot studies in order to:

- improve the design of it (3 iterations) making sure it would not affect the perceived usefulness of the commenting tool.
- make it possible to comprehend and learn in a very short time (at most 10 minutes for the complete system) in order not to keep the user from spending most of his time solving the design task.

8.2.2 Using Wizard-of-Oz techniques

When Wizard-of-Oz techniques are used, the human operator (Wizard) is the main potential source of reliability problems. Two of the most important issues are described below:

Operator expectations cause bias

Is it not possible that the results stem from the Wizard “adapting” in order to create “better” data?

A simulated system is not a real system

Can results collected using a human operator simulating a system be used to draw conclusions about a real system?

Bias due to expectations and hypotheses

In this kind of experiment, it could be the case that the experimenter acting as Wizard becomes biased due to expectations related to the hypotheses used. Of course, this may be the case even if the experimenter is not aware of it himself. There are at least two ways of avoiding this problem.

First, the experiment may be performed without a clear set of hypotheses but with a formal set of behaviour guidelines. This means that the Wizard's behaviour can still create a bias in the collected data, but that 1) the guidelines may be designed to help reduce focused bias, and 2) the behaviour is more likely to generate noise. Another solution, possibly used in combination with the guideline approach, is to use a Wizard who is not aware of the complete experiment setup and goals.

In this study, when the actual experiments were performed, we had no single set of hypotheses guiding us. Therefore, the Wizard was not biased towards a certain behaviour. The results concerning the Wizard's performance (see section 6.1) shows that he worked consistently, and without any single bias.

A simulated system is not a real system

When Wizard-of-Oz techniques are used to analyse a system, we must assure that the results would be valid for the actual system and not just the simulation. Otherwise, we cannot draw conclusions of importance for practitioners and tool-makers.

In this study, the results are valid for the simulated system. However, if the ratings of the simulated system are positive, and the simulated system is not "better" than a real system, then we might conclude that the real system would be rated at least as good. Furthermore, the conclusion is only interesting if we can show that the positive results of the simulator test do not stem from properties of the human simulating the system. Three of the systems characteristics clearly affect its ability to apply and deliver comments: knowledge, speed and perception of the situation.

Knowledge

The human operator "knows more" but is instructed not to give other comments than those in the decided set. It is technically possible to

implement a system that integrates lexical, syntactical and semantic as well as specification-related information.

Speed

The human operator is relatively slow, and probably less suited at evaluating several components in parallel. However, it is not certain that the highest speed is the ultimate solution. It might be the case that a system that is slow, and even varies in delivery speed, is preferred.

Perception of situation

The human operator interprets the situation using a multitude of channels that would not easily be available to a computerized agent. However, it is technically possible to implement a system that evaluates surface properties (see, e.g., Löwgren and Nordqvist, 1990; Mahajan and Shneiderman, 1995) as well as deeper properties (see, e.g., Weitzman, 1992; Fischer and Nakakoji, 1994) using combinations of rule-based and case-based techniques.

In our study, the human operator (Wizard) is not the primary source of the results. The focus of our appropriateness study are the dimensions mode and mood, which are more closely related to other issues: The mood dimension results depend primarily on the canned texts which were constructed before the experiment. The mode dimension results depend on the speed of the human operator in the active case. Since the operator is slower than a computerized agent would be, this could be a possible source of problems. However, as mentioned above, the critical issue is not speed, but rather timing.

8.2.3 Potential problems in the design and procedure

Of course, in retrospect, it is possible to identify several potential weak points in our experiment design, preparation and procedure. Although they do not make our conclusions less valid or reliable, we recognise them as critical issues in future experiments and continuations of this project.

Ecological validity and data size

The design sessions were too short and the design task too limited. This resulted in a very small number of comments and many similar flaws, i.e., the experiment produced rather small amounts of data for the analysis. This, and the fact that we conducted our studies in a laboratory environment, would indeed cause problems if our major goal was to draw conclu-

sions about support tools in a professional context, i.e., if ecological validity was necessary. The fact that we used students also makes the interpretation of the results more complex. For example, our observation that imperative mood is preferred by the participants may, at least to some degree, be due to the rather low design experience and limited design knowledge of the students.

However, as we discuss in chapter 3, this study is just the first step towards a more formal (and ecologically valid) theory about the effects of using knowledge-based support tools in user interface design.

Selecting a knowledge base requires better preparation

The initial guideline database should have been reviewed and tested in more depth before the actual experiment, making sure that all the guidelines were useful and applicable for the specific task. In this project, the guideline selection was conducted in a rather *ad hoc* manner, and tested only briefly, during the pilot studies. On the other hand, in a large-scale, real-world interface design project, the guideline collections are very likely to contain several guidelines that are likewise included *ad hoc*, and perhaps never used.

Potential problems in the questionnaire design

The questionnaire design may suffer from the same problems as the knowledge base construction—it should have been tested more thoroughly before the actual study. The design of the questions and scales were only iterated three times, and only tested during the rather small pilot-study, using five participants.

Important issues in this kind of study are the terminology and emphasis of words in the questionnaires and scales. Two major aspects of this are: 1) how to select terms that all participants will interpret consistently, and 2) how to select scale labels that make the scale wide enough, yet generates enough variance. In our questionnaires, we see some constructions that should have been refined further. For example:

Question 3:2, scale labels “too fast–too slow”

There are significant differences between groups for this questions, but also a clear tendency to mark centre-values (3–6). Another label (e.g. fast/slow) would probably have created larger differences. The problem with the current solution is that the small differences

(although significant) make it hard to conclude the importance of the interval.

Question c:1, scale labels “useful–not useful”

Terms like “useful”, “usable”, “helping”, etc., are difficult to interpret easily and consistently, and it is hard for us to know if all participants have interpreted them the same way. Furthermore, the Swedish term used for “usefulness” was “användbar”, which can be interpreted as both “useful” and “usable”.

What did the subjective measure of MWL really measure?

The design of the perceived MWL measurement process was problematic, and in retrospect, we want to present some questions concerning the outcome and validity of the measures.

First, the selection procedure took place during a very short period, with several other activities going on. A first selection took place during the fulfilment of the design task, as the two experimenters took notes and marked time slots of “suitable situations”. This took place when both experimenters were busy studying the participant, preparing comments, answering questions, etc. Later, at the same time as the participant filled the first questionnaire, the experimenters spent approximately 3–5 minutes discussing and filtering situations using the notes and fast replays of the video tape. There is a risk that the selection sometimes became a bit *ad hoc* or “sloppy”.

Secondly, the video replay of the situations was arranged rather informally, starting approximately 20 seconds “before” and stopping 20 seconds “after” the situation. In other words, the situation and its boundaries were relatively unclear and informally specified. This makes it harder to rely on hard comparisons (formal statistical analysis) of the situations.

Thirdly, and more importantly, what is measured depends on what the participant perceives as the essence of the “replay”. The only instructions the participants got was to 1) look at the sequence, and 2) answer the question if they recognized the situation. It is possible, although we tried to avoid it by using rather short sequences, that some sequences included several major activities and events, and that the participant interpreted it as another “situation” than the one we were interested in.

A fourth critical factor concerns the relationships between the MWL measurement interval, the video sequence interval and the participants

memory of the situation: what (MWL) is recalled when the sequence is replayed?

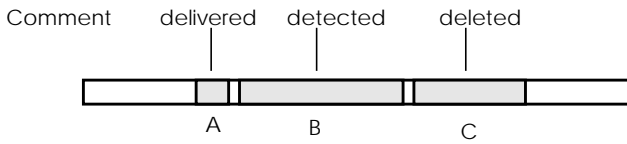


Figure 8.1 The MWL measurement interval.

In the case where a comment has been delivered, we are interested in the effects of the comment delivery, i.e., the resulting MWL. Figure 8.1 shows a simplified sequence where a comment is being delivered. In order to get a “correct” value, what should be replayed: the sequence containing the comment delivery (A), the sequence immediately afterwards (B), or both? In some cases the participant detects the comment immediately when it is delivered, but in others, the “detection phase” takes a long time. During this study, we used the approach to replay A and a short part of B.

We were aware of most of the above issues during the study, and tried to prevent problems by using 1) a consistent treatment of the participants, 2) a consistent selection and handling of the situations and video sequences, and 3) a Wizard with extensive instructions (and essentially non-biased).

8.2.4 Ethical issues—should participants be “deceived”?

Using the Wizard-of-Oz technique implies “deceiving” participants about the nature of, at least part of, the system being studied. In our experiment, the “lie” consisted of the message that the system includes a prototype expert system that is capable of commenting on your user interface design work.

It might be argued that this is unethical, and that one should never lie to a person participating in any kind of experiment; participation should be voluntary and the participants should be made fully aware of the experimental setup. This is something that is very hard to practice. Several studies, whether questionnaires, interviews or evaluation tasks, often depend on the fact that the participants are not told about the actual purpose of the study. The most obvious risk with doing so is that it might result in a biased behaviour.

Fraser and Gilbert (1991) suggest that Wizard-of-Oz experimenters should not tell an outright lie, but instead tell only half the truth, and then

let the participants draw their own conclusions. However, from a strictly ethical point of view, the difference is not that big. We believe that it is possible to argue for the use of a research method that includes deceiving if certain conditions are met:

- The participants are volunteers.
- The participants are debriefed as soon as possible after the experiment. This includes describing the actual setup, the aim of the experiment and the reasons for using this particular method.
- The participants are asked for permission to use the material, and if not, guaranteed that the data collected will be destroyed and not used in the analysis.
- The misinformation does not harm or put the participant in an embarrassing situation.

8.3 General discussion

As a by-product of our empirical study, we informally explored and addressed some issues related to the use of commenting tools as design support. We wish to present a short discussion of our experience and collected data in order to inform future research.

8.3.1 Learning effects

The CODEK project included a limited and rather informal study of the possible learning effects from the use of a commenting tool. Our intention was to find out whether or not the use of a rather *simple* environment and commenting tool could result in any long-time effects. Hence, we did not put a lot of effort in measuring more formally any information transfer or one-to-one correspondence between information received and information used later on. Among other things, that kind of study would have required a lot more participants to generate reliable results.

Learning effects would indeed be valuable if the learning process did not interfere negatively with the design and development process. This could be used both as a tool for educating new employees and project members, as well as a technique for transferring knowledge between projects and teams. For this to work, the form becomes a critical issue. How should the knowledge be presented in order to reduce the degree of disturbance and effectively educate the designer?

Previous research in the area of commenting tools have experimented with presentation-related variables such as: mood, the use of examples or

the use of rationale. Carrol et al. (1988) reports that how-to-do-it help without rationales could actually impair learning. In our limited study we identified no such effects. From a perception point-of-view, the imperative comments actually seemed to improve the system's ratings. Comments from the participants indicates that they feel that they have "learned" something from the use of the commenting tool, and in some cases also that they consider this to be useful. Some voices from the interviews after the evaluation session:

"I would not have realized that flaw if I had not used the commenting tool."

"I discovered that flaw just because I had received a comment about it last time (by the commenting tool)."

"Perhaps I was a bit more critical [during the evaluation] because I had tested your (commenting) tool."

8.3.2 Using interaction logs to predict MWL

In the CODEK project, we only performed an exploratory study of the possibility to use the designer's actions (interaction) to guide the commenting strategies. If this area is going to be studied in more depth, there are several things to be learned from our experience.

First, the basis used for interaction analysis—the components used to create the logs—should probably be selected more carefully. We opted for rather low-level events, created by adding log-callbacks to each user-accessible function in the design and commenting applications. Alternatives, not tested by us, include using, e.g.:

"Raw" mouse and keyboard events

We considered this to create too much raw data to be manageable within this study.

Abstract events, generated by mapping functions to user tasks

Higher-level events might be generated in a number of ways. A straightforward way is to map design system functions to user tasks for the already completed system. Another approach would be to build the design system with this in mind, i.e., to use a formal user-task description technique when designing the functions. This description can then be used to deduce log events. There are several task-analysis techniques (Diaper, 1989) that may be candidates for such an approach, e.g., GOMS (Card et al., 1983), CTA (Barnard, 1987), or

UAN (Hartson et al., 1990). We considered this to be too time-consuming for the main purpose of this study.

Secondly, we only looked at average quantity and speed of raw events in the interaction. There are other, less simplistic, ways to analyse the interaction. For example, one could take into account the event *density* when developing a measure of “intense activity”. Furthermore, we only looked at the presence or absence of action, i.e., counting raw events. A more sophisticated analysis of the users’ tasks and corresponding functions could be used to create *weights* for each event, e.g.: moving a single button is less “complex” than fine-tuning the location of a block of objects.

8.4 Future work

Our main goal is to make it possible to provide designers with useful and valuable support tools. In our current work, we focus on handling the problems of design information overflow and complexity. It is our intention to provide valuable information for future research projects and, eventually, commercial tool builders. This study is a first step in this direction, in that it resulted in conclusions concerning the usefulness of commenting tools and effects of different strategies. We see several ways to continue the work, making use of our current results.

A continuation could include studies of the impact on product and process (see section 3.5), as well as effects from varying other strategy variables (see section 2.5.4). However, it might also be valuable to perform a comparative study, testing different support techniques.

An important part of future activities concerns the designer in context, i.e., the tasks, environment, organisation, workflow and people in the professional design setting. In this study, we have referred to the findings and theories of others when presenting the rationale and design-related background material. In order to be able to study the issue of support tools deeper, we believe that it is necessary to investigate the professional setting. This would provide us with information about the actual problems and needs of the designer, which is essential if future tool are to be useful and valuable. There are several questions that need to be addressed in such a study:

- What are the characteristics of the information causing the problems?
- What tools are used today to support the process (and yet possibly disturb the designers)?

- To what degree must the support tools support not only personal knowledge management, but also collaboration and transfer between individuals?
- How should the support tools best be adapted to meet the needs of the design team (which may vary): designer creativity, designer learning method and guidelines, product conformance, process speed, etc.?
- What are the overall effects of a commenting tool; what about the designer's performance, the resulting artifact's usability, etc.?

In order to address these and other related questions, we identify the need to carry out studies in two different directions, and to aim at empirical studies of software prototypes in a professional setting.

8.4.1 Controlled experiments and field studies

As a first step, we plan to perform additional controlled experiments. There are more data to be analysed from this study, and several other important aspects of the design process to be investigated. Furthermore, we want to be able to express more robust conclusions—the study will include a larger set of participants, preferably with a background more similar to that of a professional designer. In this study, we only looked at two dimensions. As described in section 2.5.4, there are several strategy aspects that may have an important impact on the designer's perception as well as her work. The use of examples, visual or textual, and a more thorough treatment of the mood dimension are some of the issues we want to focus on. We also want to explore more formally and in more depth the value of using low-level data for improving delivery timing. For this kind of studies, we will continue to use a rather limited, although improved, design environment. This is partly because of the low-level data collection—we want to make use of some sort of abstract events (see section 8.3.2) and hence need access to the internal mechanisms of the environment. The controlled experiments will be performed in an expanded version of the CODEK Wizard-of-Oz environment (see chapter 4). In order to make it easier to carry the study further, we intend to perform later studies using a design environment more similar to that of a commercial UIMS or UIB.

At the same time as we want to contribute to the rather formal, qualitative knowledge about commenting tools in this domain, we also want to gain understanding about the design process, the design information inherent and handled in this process, and the use of design support tools.

The controlled experiments make it possible to evaluate presentation variations and to rule out certain strategy settings. However, to develop a prototype support tool that can be tested in a professional setting, we need to investigate the professional context and requirements.

To record and analyse information from the professional design situation, we plan to perform field studies. This will include several different types of data collection, and, if possible, take place at the same systems development company where we later intend to perform field testing of a software prototype. First, we will rely upon interviews with professional designers. The focus of the interviews will be the design knowledge and design support used in the everyday design work; we will limit the study to the detailed design and related activities. The main goal will be to identify the problems caused by or related to design information, and to try to understand the characteristics of the information causing the problems. Secondly, we want to collect information about the designers' expectations, requirements and complaints concerning the support tools. This aims at finding out what tools are used today to support the process, but also, more generally, what kind of support the designers feel that they need. We intend to collect such data via the use of focus groups, brainstorming; sessions where groups of designers are allowed to speculate and discuss their requirements regarding their design environments (and, more generally, their workplace/physical context).

During these activities, we will also try to find a project in which testing a prototype can be done.

8.4.2 Development of a prototype and field testing

The next step in the research process will be to interpret the data and results from the experiments and interviews, visualize this interpretation in an implementation and to evaluate it. We intend to develop a prototype support tool that can be used to test our results and theories. At this stage, we leave the Wizard-of-Oz environment used during the CODEK project, and, preferably, combine or integrate the prototype commenting tool in one of the design environments used by the designers. The architecture and characteristics of this prototype will of course depend on the data from the field studies, but in order to answer the kinds of questions we address, it is reasonable to assume that it will include the following components:

- A partially implemented commenting system, including, e.g.:
- A large knowledge-base, built from design knowledge used in the designers' organization. This database will probably have several "views", for testing different presentation strategies.
- A large set of logging features, for recording user behaviour. The current study showed that for informal studies and comparisons of different person's behaviour, a function for recording and playback of events is also very useful.
- A temporal strategy mechanism, based on the event recording. This mechanism will be adaptable, in order to make it possible to test, e.g., different degrees of intrusiveness.

We will strive to perform the field study and evaluation of the prototype in two different directions. First, data will be collected using interviews and focus groups. The prototype will be used to investigate and discuss the concept of a commenting tool resulting from our results and theories. The participants and evaluators will be designers from a systems development company. We believe that it will also be useful to combine this with a set of controlled studies, similar to those reported here, of the same environment. Data will be collected using observation, interviews and the environment's event recording mechanisms.

Secondly, we will extend the scope from the CODEK project and investigate the impact on the actual process and product too. We want to be able to answer questions concerning the designer's performance in terms of speed and "accuracy", but also regarding her creativity and ability to collaborate with her colleagues. From a product-oriented perspective, we also intend to collect data about the resulting artifact's usability. We believe that this issue can be explored by letting a small set of designers use the prototype to solve a "realistic" design task (possibly from a previous project not involving these designers), and then evaluating the results.

8.4.3 Expected results and contributions

What is the goal of the studies proposed here? We hope to make at least three contributions to the tool developers and researchers in the domain of design support tools.

First, we aim at gaining an understanding of the use of design support tools during detailed design; requirements and user expectations grounded in preferences, performance measurements and product aspects. Hope-

fully, this can guide us and other researchers when proposing and testing support techniques and new support tools.

Secondly, we intend to identify the general use of design knowledge, and the possible attempts to transfer such knowledge between designers/developers and projects/processes. From a practical point-of-view, we hope to isolate problems experienced due to the use of guidelines and written design information, in order to introduce better support techniques. However, it is also our intention to be able to reason about design knowledge in practice on a more general level, not just in the perspective of today's information overflow and complexity.

Thirdly, we want information about the value of using a commenting tool as part of a design support environment. This will be the final stage of the studies introduced in the CODEK project. Our intention is to achieve results that can be used to rate such tools in terms of user preferences, performance, product "quality", and process impact. This would serve as guidance for tool developers as well as researchers. A very practical, potential, outcome of the prototype development and field studies will be the prototype commenting tool. In case the tool is rated valuable, both experimentally and in practice, the software company will have the basis for developing a support tool of their own.

8.4.4 Long-term goals—comparative studies

In a longer perspective, we want to widen the scope of the support tool studies; we are also interested in comparative studies, where effects of alternative techniques may be evaluated. As mentioned in section 2.4, there are many potential support technologies (cf. Bias et al., 1993). It is our intention to use the field studies as a means of collecting information about the general support needs of the designers, not just the expectations related to commenting or guideline-type information. We hope to collect enough data to be able to identify a couple of viable alternatives to the commenting approach. Some potential techniques are:

- constraints in tools/libraries
- hypertext databases
- automatic design/refinement
- (electronic) gatekeepers
- printed material
- education/training

The main part of the continued work would be a comparative study of different support techniques, used in a professional software development context.

References

- Alslys Inc. (1994). *TeleUSE System Overview*. Alslys, Inc. San Diego, CA.
- Apple Computer (1987). *Apple Human Interface Guidelines*. Reading MA: Addison-Wesley.
- Balbo, S., Coutaz, J., and Salber, D. (1993). Towards automatic evaluation of multimodal user interfaces. In *Proc. of the 1993 International Workshop on Intelligent Interfaces, Orlando, Florida*, pages 201–208.
- Barnard, P. J. (1987). Cognitive resources and the learning of human-computer dialogs. In Carroll, J., editor, *Interfacing Thought, Cognitive Aspects of Human-Computer Interaction*, pages 112–158. MIT Press.
- Bellotti, V. (1988). Implications of current design practice for the use of HCI techniques. In Jones, D. M. and Winder, R., editors, *Proc. of the HCI'88 Conference on People and Computers V*, pages 13–34. Cambridge University Press.
- Bereiter, C. and Scardamalia, M. (1989). Intentional learning as a goal of instruction. In Resnick, L. B., editor, *Knowing, Learning and Instruction: Essays in Honor of Robert Glaser*. Lawrence Erlbaum Associates.
- Berlage, T. (1991). The GINA inteface builder. In *First Annual International Motif Users Meeting, Washington D.C.*
- Bias, R., Gillan, D. J., and Tullis, T. S. (1993). Three usability enhancements to the human factors-design interface. In Salvendy, G. and Smith, M. J., editors, *Proc. of the 5th International Conference on Human-Computer Interaction (HCI International'93)*, pages 169–174. Elsevier Science.
- Birnbrauer, H. (1987). Evaluation techniques that work. *Training and Development Journal*, July issue, pages 53–55.
- Blatt, L. A. and Zacherl, A. (1993). User interface requirements for the representation of examples in a user interface design guidance system. In Wiklund, M. E., editor, *Proc. of ACM INTERCHI'93 Conference on Human Factors in Computing Systems – Adjunct Proceedings*, pages 31–32.
- Bodart, F., Hennebert, A.-M., Lehereux, J.-M., and Vanderdonckt, J. (1994). Towards a dynamic strategy for computer-aided visual place-

- ment. In *AVI'94, 2nd Workshop on Advanced Visual Interfaces*. ACM Press.
- Bodart, F. and Vanderdonckt, J. (1995). Using ergonomic rules for evaluation by linguistic ergonomic criteria. In *Advances in Human Factors/ Ergonomics Series, Vol. 20B Symbiosis of Human and Artifact: Human and Social Aspects of Human-Computer Interaction, Proc. of the 6th International Conference on Human-Computer Interaction (HCI International'95)*, pages 367–372. Elsevier Science.
- Bonnardel, N. and Sumner, T. (1994). From system development to system assessment: Exploratory study of the activity of professional designers. In *Proc. of ECCE'7: 7th European Conference on Cognitive Ergonomics (GMD-studien Nr. 233)*, pages 23–36.
- Brown, A. L. and Palincsar, A. S. (1989). Guided, cooperative learning and individual knowledge acquisition. In Resnick, L. B., editor, *Knowing, Learning and Instruction: Essays in Honor of Robert Glaser*. Lawrence Erlbaum Associates.
- Brown, C. M. (1988). *Human-Computer Interface Design Guidelines*. Ablex Publishing.
- Bryman, A. (1988). *Quantity and Quality in Social Research*, (edited by Bulmer, M.), volume 18 of *Contemporary Social Research*. Routledge.
- Byers, J. C., Bittner Jr., A. C., and Hill, G. S. (1989). Traditional and raw task load index (TLX) correlations: Are paired comparisons necessary? In Mital, A., editor, *Advances in Industrial Ergonomics and Safety*. I. Taylor & Francis.
- Byrne, M. D., Wood, S. D., Sukaviriya, P., Foley, J. D., and Kieras, D. E. (1994). Automating interface evaluation. In *Human Factors in Computing Systems (CHI'94 Proceedings)*, pages 232–237.
- Card, S. K., Moran, T. P., and Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates.
- Carey, T. (1988). The gift of good design tools. In Hartson, H. R. and Hix, D., editors, *Advances in Human-Computer Interaction*, chapter 5.
- Carroll, J. M. and Aaronson, A. P. (1988). Learning by doing with simulated intelligent help. *Communications of the ACM*, 31(9):1064–1079.
- Chapanis, A. (1990). Specifying human computer interface requirements.

- Behaviour and Information Technology*, 9(6):479–492.
- Cohen, A. (1995). A description of MIL-STK-1472. *SIGCHI Bulletin*, 27(2):48–49. Report from a CHI'94 Special Interest Group: Tools for Working With Guidelines.
- Conway, M., Pausch, R., and Passarella, K. (1992). A tutorial for SUIT. Technical report, Computer Science Department, University of Virginia.
- Coutaz, J. (1994). Evaluation techniques: Exploring the intersection of HCI and software engineering. In Taylor, R. N. and Coutaz, J., editors, *Lecture notes in Computer Science, vol 896, Proc. of International Workshop on Software Engineering and Human-Computer Interaction (ICSE'94 Workshop on SE-HCI), Sorrento, Italy, May16-17*.
- Coutaz, J., Salber, D., and Balbo, S. (1993a). Towards automatic evaluation of multimodal user interfaces. *Knowledge-Based Systems Journal*, 6(4):267–274.
- Coutaz, J., Salber, D., and Balbo, S. (1993b). Towards automatic evaluation of multimodal user interfaces. Technical Report Amodeus Project Document: SM/WP32, The AMODEUS Project.
- Dahlbäck, N. (1992). *Representation of Discourse: Cognitive and Computational Aspects*. PhD thesis Linköping Studies in Science and Technology #264, Linköping University, Institute of Technology, Sweden.
- Dahlbäck, N., Jönsson, A., and Ahrenberg, L. (1993). Wizard of Oz-studies – why and how. In *Workshop on Intelligent User Interfaces*, Orlando, FL.
- de Souza, F. and Bevan, N. (1990). The use of guidelines in menu interface design: Evaluation of a draft standard. In *Proc. of IFIP Interact'90*, pages 435–440. Elsevier Science.
- Desmarais, M. C., Hayne, C., Jagannath, S., and Keller, R. (1994). A survey on user expectations for interface builders. In *Human Factors in Computing Systems (CHI'94 Proceedings)*, pages 279–280. Short paper.
- Diaper, D. (1989). *Task Analysis for Human-Computer Interaction*. Ellis Horwood.
- Dilli, I. and Hoffman, H.-J. (1995). Diades-II – a multi-agent user interface design approach with an integrated assessment component. *SIGCHI*

- Bulletin*, 27(2):33–34. Report from a CHI'94 Special Interest Group: Tools for Working With Guidelines.
- DoD (1974). MIL-STD-1472B, military standard: Human engineering design criteria for military systems, equipment and facilities.
- Eberts, R. E. (1994). *User Interface Design*. Prentice-Hall.
- Edmonds, E. (1994). Introduction: Computer-based systems that support creativity. In Dartmall, T., editor, *Artificial Intelligence and Creativity*, pages 327–334. Kluwer Academic Publishers.
- EEC (1990). Directive concerning the minimum safety and health requirements for VDT workers. 90/270/EEC.
- Eriksson, H., Frost, N., and Musen, M. A. (1996). Design critiquing for a knowledge-engineering development environment. In *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Alberta, Canada, November 9-14, 1996*. to be published.
- Fischer, G. (1991). Supporting learning on demand with design environments. In *Proceedings of the International Conference on the Learning Sciences*, pages 165–172.
- Fischer, G. (1994). Putting the owners of problems in charge with domain-oriented design environments. In Gilmore, D. J., Linder, R. L., and D tienne, F., editors, *User-Centered Requirements for Software Engineering Environments*, pages 297–306. Springer-Verlag.
- Fischer, G., Lemke, A., Mastaglio, T., and Morch, A. (1990). Using critics to empower users. In *Human Factors in Computing Systems (CHI'90 Proceedings)*, pages 337–347, Seattle.
- Fischer, G., Lemke, A. C., and Mastaglio, T. (1991a). Critics: An emerging approach to knowledge-based human-computer interaction. *International Journal of Man-Machine Studies*, 35(5):695–721.
- Fischer, G., Lemke, A. C., Mastaglio, T., and Morch, A. I. (1991b). The role of critiquing in cooperative problem solving. *ACM Transactions in Information Systems*, 9(3):123–151.
- Fischer, G. and McCall, R. (1989). Janus: Integrating hypertext with a knowledge-based design environment. In *Proc. of HyperText'89*, pages 105–117.
- Fischer, G., McCall, R., and Morch, A. (1989). Design environments for

- constructive and argumentative design. In *Human Factors in Computing Systems (CHI'89 Proceedings)*, pages 269–275.
- Fischer, G. and Nakakoji, K. (1991). Making design objects relevant to the task at hand. In *Proc. of 9th National Conference on AI (AAAI'91)*, pages 67–73. AAAI Press/MIT Press, Cambridge, MA.
- Fischer, G. and Nakakoji, K. (1992). Beyond the macho approach of artificial intelligence: Empower human designers – do not replace them. *Knowledge-Based Systems Journal*, 5(2):15–30.
- Fischer, G. and Nakakoji, K. (1994). Amplifying designers' creativity with domain-oriented design environments. In Dartmall, T., editor, *Artificial Intelligence and Creativity*, pages 343–364. Kluwer Academic Publishers.
- Fischer, G., Nakakoji, K., and Ostwald, J. (1995). Supporting the evolution of design artifacts with representations of context and intent. In *Symposium on Designing Interactive Systems (DIS'95 Proceedings)*. ACM Press.
- Fischer, G., Nakakoji, K., Ostwald, J., Stahl, G., and Sumner, T. (1993a). Embedding critics in design environments. *The Knowledge Engineering Review*, 8(4):285–307.
- Fischer, G., Nakakoji, K., Ostwald, J., Stahl, G., and Sumner, T. (1993b). Embedding critics in design environments. In *Human Factors in Computing Systems (INTERCHI'93 Proceedings)*, pages 157–164. New York: ACM Press.
- Fox, J., editor (1993). *Knowledge Engineering Review: Special Issue on Critiquing*. Number 4. Cambridge University Press.
- Fox, J. A. (1992). The effects of using a hypertext tool for selecting design guidelines. In *Proc. of the Human Factors Society 36th annual meeting (HFS'92)*, pages 428–432.
- Francony, J.-M., Kuijpers, E., and Polity, Y. (1992). Towards a methodology for Wizard-of-Oz experiments. In *Proc. of 3rd Conference on Applied Natural Language Processing*.
- Fraser, N. and Gilbert, N. S. (1991). Simulating speech systems. *Computer Speech and Language*, 5:81–99.
- Gillan, D. J. and Bias, R. G. (1992). The interface between human factors and design. In *Proc. of the Human Factors Society 36th annual meet-*

- ing (*HFS'92*), pages 443–447.
- Greeno, J. G. (1989). Situations, mental models, and generative knowledge. In Klahr, D. and Kotowsky, K., editors, *Complex Information Processing: The Impact of Herbert Simon*, chapter 11, pages 285–318. Lawrence Erlbaum Associates.
- Guerlain, S. A. and Smith, P. J. (1993). The role of the computer in team problem-solving: Critiquing or partial automation. In *Proc. of the Human Factors and Ergonomics Society 37th annual meeting (HFES'93)*.
- Guindon, R. (1988). How to interface to advisory systems? Users request help with a very simple language. In *Human Factors in Computing Systems (CHI'88 Proceedings)*, pages 191–196.
- Hammond, N., Gardiner, M., Christie, B., and Marshall, C. (1987). The role of cognitive psychology in user-interface design. In Gardiner, M. and Christie, M., editors, *Applying Cognitive Psychology to User-Interface Design*, chapter 2, pages 13–53. John Wiley & Sons.
- Harstad, B. (1993). New approaches for critiquing systems: Pluralistic critiquing, consistency critiquing, and multiple intervention strategies. CU-CS-685-93, Dept. of Computer Science, University of Colorado at Boulder.
- Hart, S. G. and Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In Hancock, P. S. and Meshkati, N., editors, *Human Mental Workload*, pages 139–183. Amsterdam: North Holland.
- Hartson, H. and Boehm-Davis, D. (1993). User interface development processes and methodologies. *Behaviour and Information Technology*, 12(2):98–114.
- Hartson, H. R. and Hix, D. (1989). Toward empirically derived methodologies and tools for human-computer interface development. 31(4):477–494.
- Hartson, H. R., Siochi, A. C., and Hix, D. (1990). The UAN: A user-oriented representation for direct manipulation interface designs. *ACM Transactions on Information Systems*, 8.
- Hauptmann, A. G. (1989). Speech and gestures for graphic image manipulation. In *Human Factors in Computing Systems (CHI'89 Proceed-*

- ings), pages 241–245.
- Henninger, S., Haynes, K., and Reith, M. (1995). A framework for developing experience-based usability guidelines. In *Symposium on Designing Interactive Systems (DIS'95 Proceedings)*, pages 45–53. New York: ACM Press.
- Hill, W. (1993). A Wizard-of-Oz study of advice giving and following. *Human-Computer Interactions*, 8(1):57–81.
- Hix, D. (1986). Assessment of an interactive environment for developing human-computer interfaces. In *Proc. of the Human Factors Society 30th annual meeting (HFS'86)*, pages 1349–1353.
- Hix, D. (1989). A procedure for evaluating human-computer interface development tools. In *Proc. of the 1989 ACM SIGGRAPH Symposium on User Interface Software and Technology (UIST'89)*, pages 53–61.
- Hix, D. and Ryan, T. (1992). Evaluating user interface development tools. In *Proc. of the Human Factors Society 36th annual meeting (HFS'92)*, pages 374–378.
- Hix, D. and Schulman, R. S. (1991). Human-computer interface development tools: A methodology for their evaluation. 34(3):75–87.
- Hägglund, S. (1993). Introducing expert critiquing systems. *The Knowledge Engineering Review*, 8(4):281–284.
- Iannella, R. (1994). Hypersam: A practical user interface guidelines management system. In *2nd Annual CHISIG (Queensland) Symposium, Bond University, Australia*.
- Iannella, R. (1995). HyperSAM – a management tool for large user interface guideline sets. *SIGCHI Bulletin*, 27(2):42–44. Report from a CHI'94 Special Interest Group: Tools for Working With Guidelines.
- ISO (1992). ISO 9241: Ergonomic requirements for office work with visual display terminals (VDTs). Technical report, ISO/IEC Copyright Office, Geneva, Switzerland. Draft versions parts 1, 2, 10.
- Jeffries, R., Miller, J., Wharton, C., and Uyeda, K. (1991). User interface evaluation in the real world: A comparison of four techniques. In *Human Factors in Computing Systems (CHI'91 Proceedings)*, pages 119–24. ACM.
- Jiang, J., Murphy, E. D., Bailin, S. C., and Truszkowski, W. F. (1992). Automating a human factors evaluation of graphical user interfaces for

- NASA applications: an update on CHIMES. In *SpaceOps '92: Proceedings of the Second International Symposium on Ground Data Systems for Space Mission Operations*, Pasadena, CA. Jet Propulsion Laboratory.
- Johnson, J. A., Nardi, B. A., Zarnier, C. L., and Miller, J. R. (1993). ACE: Building interactive graphical applications. *Communications of the ACM*, 36(4):41–55.
- Jönsson, A. and Dahlbäck, N. (1988). Talking to a computer is not like talking to your best friend. In *Proc. of SCAI88 - 1st Scandinavian Conference on Artificial Intelligence*. Amsterdam: IOS.
- Keller, R. K. (1994). User interface tools: A survey and perspective. In *Lecture notes in Computer Science, vol 896, Proc. of International Workshop on Software Engineering and Human-Computer Interaction (ICSE'94 Workshop on SE-HCI), Sorrento, Italy, May16-17*, pages 225–231. Springer-Verlag.
- Kim, W. C. and Foley, J. D. (1993). Providing high-level control and expert assistance in the user interface presentation design. In *Human Factors in Computing Systems (INTERCHI'93 Proceedings)*, pages 430–437. New York: ACM Press.
- Kirkpatrick, Donald, L. (1979). Techniques for evaluation training programs. *Training and Development Journal*.
- Kluger, A. N., Adler, S., and Fay, C. (1991). Computerized feedback effects on feedback seeking, performance and motivation. In Nurminen, M. I., Järvinen, P., and Weir, G., editors, *Human Jobs and Computer Interfaces Conference*, pages 131–146. Amsterdam: North Holland.
- Koivunen, M.-R., Lassila, O., Ahvo, J., Rankinen, M., Riihiäho, S., and Riihinen, B. (1993). ActorStudio: An interactive user interface editor. In Grechenig, T. and Tscheligi, editors, *Proc. of Vienna Conference, Human Computer Interaction. (VHCI'93)*. Springer-Verlag.
- Kolski, C. and Millot, P. (1991). A rule-based approach to the ergonomic “static” evaluation of man-machine graphic interface in industrial processes. *International Journal of Man-Machine Studies*, 25:657–674.
- Landauer, T. K. (1987). Psychology as a mother of invention. In *Human Factors in Computing Systems and Graphics Interface (CHI+GI'87 Proceedings)*, pages 284–298.

- Langlotz, C. and Shortcliffe, E. (1983). Adapting a consultation system to critique user plans. *International Journal of Man-Machine Studies*, 19:479–496.
- Lave, J. (1988). *Cognition in Practice*. Cambridge University Press.
- Lemke, A. and Fischer, G. (1990). A cooperative problem solving system for user interface design. In *Eight National Conference on Artificial Intelligence, AAAI-90*, pages 479–484. AAAI Press/The MIT Press.
- Löwgren, J. (1991). *Knowledge-Based Design Support and Discourse Management Systems*. PhD thesis Linköping Studies in Science and Technology #239, Linköping University, Institute of Technology, Sweden.
- Löwgren, J. (1995). Applying design methodology to software development. In *Symposium on Designing Interactive Systems (DIS'95 Proceedings)*, pages 87–95. ACM Press.
- Löwgren, J. and Laurén, U. (1993). Supporting the use of guidelines and style guides in professional user-interface design. *Interacting with Computers*, 5(4):385–396.
- Löwgren, J. and Nordqvist, T. (1990). A knowledge-based tool for user interface evaluation and its integration in a UIMS. In Diaper, D., Gilmore, D., Cockton, G., and Schackel, B., editors, *Human-Computer Interaction – Interact'90*, pages 395–400, North-Holland. Also as research report LiTH-IDA-R-90-15, Dept. of Computer and Information Science, Linköping University, Sweden.
- Löwgren, J. and Nordqvist, T. (1992). Knowledge-based evaluation as design support for graphical user interfaces. In *Human Factors in Computing Systems (CHI'92 Proceedings)*, pages 181–188. New York: ACM Press.
- Luotonen, A., Nielsen, H. F., and Berners-Lee, T. (1996a). CERN httpd. Technical report, CERN. URL: <http://www.w3.org/pub/WWW/Daemon/>.
- Luotonen, A., Nielsen, H. F., and Berners-Lee, T. (1996b). W3C httpd CGI/1.1 script support. Technical report, CERN. URL: <http://www.w3.org/pub/WWW/Daemon/User/CGI/>.
- Mahajan, R. and Shneiderman, B. (1995). A family of user interface consistency checking tools. Technical Report CAR-TR-770, Human-

- Computer Interaction Laboratory, Department of Computer Science
University of Maryland College Park.
- Malinowski, U. and Nakakoji, K. (1995). Using computational critics to facilitate long-term collaboration in user interface design. In *Human Factors in Computing Systems (CHI'95 Proceedings)*, pages 385–392.
- Marcus, A. (1991). *Graphic Design for Electronic Documents and User Interfaces*. New York: ACM Press.
- Maulsby, D., Greenberg, S., and Mander, R. (1993). Prototyping an intelligent agent through Wizard-of-Oz. In *Human Factors in Computing Systems (INTERCHI'93 Proceedings)*, pages 277–284.
- Microsoft Corporation (1992). *The Windows Interface: An Application Design Guide*. Redmond WA:Microsoft Press.
- Miller, P. L. (1983). Attending: Critiquing a physician's management plan. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(5).
- MIT Media Lab (1994). Design interaction paradigms home page. WWW document. URL: <http://design-paradigms.www.media.mit.edu/projects/design-paradigms/>.
- Morris, N. M. (1987). Designing for user acceptance of design aids. In Rouse, W. B. and Boff, K. R., editors, *System Design: Behavioural Perspectives on Designers, Tools, and Organizations*, pages 245–255.
- Mosier, J. N. and Smith, S. L. (1986). Application of guidelines for designing user interface software. *Behaviour and Information Technology*, 5(1):39–46.
- Muckler, F. A. and Seven, S. A. (1992). Selecting performance measures: "objective" versus "subjective" measurement. *Human Factors*, 34(4):441–455.
- Myers, B. (1994). Challenges of HCI design and implementation. *Interactions*, 1(1):73–80.
- Nakakoji, K. and Fischer, G. (1995). Intertwining knowledge delivery and elicitation: A process model for human-computer collaboration in design. *Knowledge-Based Systems Journal*, 8(2-3):95–104.
- Nakakoji, K., Malinowski, U., and J., L. (1996). Knowledge-based support for the user-interface design process: A CHI95 workshop. *SIGCHI Bulletin*, 28(1):43–47.

- Nakakoji, K., Reeves, B., Aoki, A., Suzuki, H., and Mizushima, K. (1995). eMMaC: Knowledge-based color critiquing support for novice multimedia authors. In *Proc. of ACM Multimedia'95*, pages 467–476. ACM Press.
- Nakakoji, K., Sumner, T. R., and Harstad, B. (1994). Perspective-based critiquing: Helping designers cope with conflicts among design intentions. In Gero, J. and Sudweeks, F., editors, *Proc. of AI in Design'94*, pages 449–466. Kluwer Academic Publishers.
- NASA (1992). Human-computer interface guidelines. Technical Report DSTL-92-007, National Aeronautics and Space Administration, Goddard Space Flight Center, Greenbelt Maryland.
- Neches, R., Foley, J., Szekely, P., Sukaviriya, P., Luo, P., Kovacevic, S., and Hudson, S. (1993). Knowledgeable development environments using shared design models. In *Proc. of the 1993 International Workshop on Intelligent Interfaces, Orlando, Florida*, pages 63–70. New York: ACM Press.
- Nielsen, J., editor (1989). *Coordinating User Interfaces for Consistency*. Academic Press.
- Nielsen, J. and Molich, R. (1989). Teaching user interface design based on usability engineering. *SIGCHI Bulletin*, 21(1):45–48.
- Ogawa, K. (1993). The role of design guidelines in assisting the interface design task. In *Proc. of the Human Factors and Ergonomics Society 37th annual meeting (HFES'93)*, pages 272–276.
- Ogawa, K. and Ueno, K. (1995). GuideBook – design guidelines database for assisting the interface design task. *SIGCHI Bulletin*, 27(2):38–39. Report from a CHI'94 Special Interest Group: Tools for Working With Guidelines.
- Olsen, Jr., D. R. and Halversen, B. W. (1988). Interface usage measurements in a user interface management system. In *ACM SIGGRAPH Symposium on User Interface Software. UIST'88*, pages 103–108. ACM.
- OSF (1991). *Motif Style Guide*. Open Software Foundation.
- Perlman, G. (1988). Software tools for user interface development. In Helander, M., editor, *Handbook of Human-Computer Interaction*, chapter 37, pages 819–833. Elsevier Science.

- Perlman, G. (1989). Asynchronous design/evaluation methods for hyper-text technology development. In *Proc. of HyperText'89*, pages 61–81.
- Perlman, G. and Moorhead, A. J. (1988). Applying hypertext methods for the effective utilization of standards. In *IEEE COMPSTAN'88 Conference on Computer Standards*.
- Philips, B. H. (1993). Developing interactive guidelines for software user interface design: A case study. In *Proc. of the Human Factors and Ergonomics Society 37th annual meeting (HFES'93)*, pages 263–267.
- Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., and Carey, T. (1994). *Human-Computer Interaction*. Addison-Wesley Publishing Company.
- Puerta, A. R., Eriksson, H., Gennari, J. H., and Musen, M. A. (1994). Beyond data models for automated ui generation. In *Proc. of the HCI'94 Conference on People and Computers IX*, pages 353–366. Cambridge University Press.
- Reaux, R. A. and Willeges, R. C. (1988). Effects of abstraction and presentation on usability of user-system interface guidelines. In *Proc. of the Human Factors Society 32nd annual meeting (HFS'88)*, pages 330–334.
- Reder, L. M. and Ritter, F. E. (1992). What determines initial feeling of knowing? Familiarity with question terms, not with the answer. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 18(2).
- Redmiles, D. F. (1995). Observations on using empirical studies in developing a knowledge-based software engineering tool. Technical report, Department of Computer Science and Institutie of Cognitive Science, University of Colorado, Boulder.
- Reiterer, H. (1993). A human factors based user interface design. In Grechenig, T. and Tscheligi, M., editors, *Proc. of Vienna Conference, Human Computer Interaction. (VCHCI'93)*. Springer-Verlag.
- Reiterer, H. (1994). Design aid tools for user interface designers. In *Proc. of ECCE'7: 7th European Conference on Cognitive Ergonomics (GMD-studien Nr. 233)*, pages 325–339.
- Reiterer, H. (1995). IDA – user interface design assistance. *SIGCHI Bulletin*, 27(2):46–47. Report from a CHI'94 Special Interest Group:

- Tools for Working With Guidelines.
- Rittel, A. and Webber, M. M. (1984). Planning problems are wicked problems. In Cross, N., editor, *Developments in Design Methodology*, pages 134–144.
- Rosson, M. B., Maass, S., and Kellogg, W. A. (1987). Designing for designers: An analysis of design practices in the real world. In *Human Factors in Computing Systems and Graphics Interface (CHI+GI'87 Proceedings)*, pages 137–142.
- Rosson, M. B., Maass, S., and Kellogg, W. A. (1988). The designer as user: Building requirements for design tools from design practice. *Communications of the ACM*, 31(11):1288–1297.
- Salber, D. and Coutaz, J. (1993). Applying the Wizard-of-Oz technique to the study of multimodal systems. In Bass, L., Gornostaaev, J., and Unger, C., editors, *Lectures in Computer Science, vol 753, Proc. of HCI, 3rd International Conference EWCHI'93, East/West Human Computer Interaction, Moscow*. Springer-Verlag.
- Schön, D. (1983). *The Reflective Practitioner*. Basic Books.
- Sears, A. (1993). Layout appropriateness: A metric for evaluating user interface widget layouts. *IEEE Transactions on Software Engineering*, 19(7):707–719.
- Sears, A. (1995). AIDE: A step toward metric-based interface development tools. In *Proc. of the 1995 ACM SIGGRAPH Symposium on User Interface Software and Technology (UIST'95)*.
- Shipman III, F. M. and McCall, R. (1994). Supporting knowledge-based evolution with incremental formalization. In *Human Factors in Computing Systems (CHI'94 Proceedings)*, pages 285–291.
- Shneiderman, B. (1992). *Designing the User Interface*. Addison-Wesley Publishing Company.
- Shneiderman, B., Chimera, R., Jog, N., Stimart, R., and White, D. (1995). Evaluating spatial and textual style of displays. Technical Report CAR-TR-763, Human-Computer Interaction Laboratory, Department of Computer Science University of Maryland College Park.
- Silverman, B. G. (1992a). *Critiquing Human Error: A Knowledge Based Human-Computer Collaboration Approach*. Academic Press.
- Silverman, B. G. (1992b). Evaluating and refining expert critics. *Decision*

- Sciences*, 23(1):86–110.
- Silverman, B. G. (1992c). Survey of expert critiquing systems: Practical and theoretical frontiers. *Communications of the ACM*, 35(4):106–127.
- Silverman, B. G. and Mazher, T. M. (1992). Expert critics in engineering design: Lessons learned and research needs. *AI Magazine*, 13(1):45–62.
- Simon, H. A. (1981). *The Sciences of the Artificial*. MIT Press.
- Smith, S. L. (1988). Standards versus guidelines for designing user interface software. In Helander, M., editor, *Handbook of Human-Computer Interaction*, chapter 40, pages 877–889.
- Smith, S. L. and Mosier, J. N. (1986). Guidelines for designing user interface software. Technical Report ESD-TR-86-278, Mitre Corp., Bedford, MA.
- Stolze, M. (1994). Visual critiquing in domain-oriented design environments: Showing the right thing at the right place. In *Proc. of AI in Design'94*.
- Stone, D. K., Sharp, H. C., and Benyon, D. R. (1993). Intelligent support for human-computer interaction design. In Graham, I. M., editor, *Proc. of Expert Systems'93. The 13th Annual Conf. of the British Computer Society Specialist Group on Expert Systems*.
- Streveler, D. and Wasserman, A. (1987). Quantitative measures of the spatial properties of screen designs. In *Proc. of IFIP Interact'87*, pages 125–133.
- Suchman, L. A. (1987). *Plans and Situated Actions*. Cambridge University Press.
- Sukaviriya, P., Foley, J. D., and Todd, G. (1993). A second generation user interface design environment: The model and the runtime architecture. In *Human Factors in Computing Systems (INTERCHI'93 Proceedings)*, pages 375–382.
- Sukaviriya, P. N. and Foley, J. D. (1990). Coupling a UI framework with automatic generation of context-sensitive animated help. In *Proc. of UI'90*, pages 152–166.
- Sumner, T. (1995). The high-tech toolbelt: A study of designer in the workplace. In *Human Factors in Computing Systems (CHI'95 Pro-*

- ceedings*), pages 178–185.
- Sun Microsystems, Inc. (1991). *OpenWindows Developer's Guide: User's Guide*. Sun Microsystems Inc.
- Sweeney, M., Maguire, M., and Shackel, B. (1993). Evaluating user-computer interaction: a framework. *International Journal of Man-Machine Studies*, 38(4):689–711.
- Szekely, P., Luo, and Neches, R. (1993). Beyond interface builders: Model-based interface tools. In *Human Factors in Computing Systems (INTERCHI'93 Proceedings)*, pages 383–390.
- Tetzlaff, L. and Schwartz, D. R. (1991). The use of guidelines in interface design. In *Human Factors in Computing Systems (CHI'91 Proceedings)*, pages 329–333.
- Thovtrup, H. and Nielsen, J. (1991). Assessing the usability of a user interface standard. In *Human Factors in Computing Systems (CHI'91 Proceedings)*, pages 335–41. ACM New York, NY, USA.
- Tullis, T. S. (1984). A computer-based tool for evaluating alphanumeric displays. In Shackel, B., editor, *Proc. of IFIP Interact'84*, pages 719–723. Elsevier Science.
- Tullis, T. S. (1988). Screen design. In Helander, M., editor, *Handbook of Human-Computer Interaction*, chapter 18, pages 377–411.
- Vanderdonckt, J. (1993). A corpus of selection rules for choosing interaction objects. Technical Report 93/3, Institut d'Informatique, Facultes Universitaires Notre-Dame de la Paix, Namur.
- Vanderdonckt, J. (1995). Accessing guidelines information with Sierra. In Nordbyn, K., Helmersen, P., Gilmore, D., and Arnesen, S., editors, *Proc. of IFIP Interact'95*, pages 311–316. London: Chapman & Hall.
- Vanderdonckt, J. and Gillo, X. (1994). Visual techniques for traditional and multimedia layouts. In T. Catarci, M.F. Costabile, S. L. G. S., editor, *AVI'94, 2nd Workshop on Advanced Visual Interfaces*, pages 95–104. ACM Press.
- Waern, Y. and Ramberg, R. (1990). Do people trust computers as experts? Technical Report Hufacit 22, Dept. of Psychology, University of Stockholm.
- Weitzman, L. (1992). Designer: A knowledge-based graphic design assistant. In *Proc. of AI in Engineering Design*, volume 1, pages 435–466.

- Wiecha, C. and Boies, S. (1990). Generating user interfaces: principles and use of ITS style rules. In *UIST. Third Annual Symposium on User Interface Software and Technology. Proceedings of the ACM SIGGRAPH Symposium*, pages 21–30. ACM New York, NY, USA.
- Wilson, J. and Rosenberg, D. (1988). Rapid prototyping for user interface design. In Helander, M., editor, *Handbook of Human-Computer Interaction*, pages 859–875.
- Wilson, S., Johnson, P., Kelly, C., Cunningham, J., and Markopoulos, P. (1993). Beyond hacking: A model-based approach to user interface design. In *Proc. of the 5th International Conference on Human-Computer Interaction (HCI International'93)*, pages 217–231.
- Winograd, T. (1995). From programming environments to environments for designing. *Communications of the ACM*, 38(6):65–74.
- Winograd, T. and Flores, F. (1986). *Understanding Computers and Cognition: A New Foundation for Design*. Ablex Publishing Company.

Appendix A

Design-task material

A.1 Written task specification

The following text (in Swedish) was used to describe the design task (see sections 3.5 and 5.5).

“Desktop Hotel Info”

You are an interface designer in a middle-sized software development company. Your task is to design the interface for the application “Desktop Hotel Info”, which is to be used for searching information before reserving hotel rooms.

Usage domain

“Desktop Hotel Info” will be used by people who are going on business trips. The idea is to make it trip. The application should only cover travel within Sweden.

The user should be able to search for suitable hotels, print the resulting information, and then deliver it to a secretary, who takes care of the actual reservation.

The intended user group use SUN workstations, a UNIX-like operative system, and the Motif window environment.

Usage

The user should be able to search for hotels matching the intentions and requirements she has. When the application has found a number of hotels matching the user’s search request, the resulting information should be presented. If no matching hotels are found, the user should be informed about this.

Input

The following information should be used to define the user’s demands:

- Location (one of many names of cities; possibly entered manually)
- Period (a date interval)
- Number of two-bed rooms
- Number of single-bed rooms
- Price class (upper price limit)
- Standard (3-, 4-, or 5-star)
- Pool (yes or no)

The user should not have to enter all the information described above. For those fields where no information has been entered, all possible values should be matched.

Output

The application communicates with a database containing all the information you will need, for all hotels in Sweden. You do not have to handle the communication.

When the application has found a number of hotels matching the user's search request, the resulting information should be presented. The user should have access to the following two types of information:

- An overview of the search results (the matching hotels)
- A detailed description for one hotel at the time

Overview

In the overview, the following information should be included:

- Hotel name
- Location

Detailed description

In the detailed description, the following information should be included:

- Name
- Address
- Zip code and city

Appendix B

Questionnaires

B.1 Questions about the design environment

#	Question	Scale endpoints (1–8)	
2:1	How would you describe your work on trying to solve the task?	pleasant	stressful
2:2	How hard did you work in order to produce a solution for this task?	not hard at all	very hard
2:3	During the experiment, were you ever unsure of what you were supposed to do?	no, not at all	yes, very much
2:4	How would you describe the difficulty of the task?	very simple	very hard
2:5	Do you believe that you will work on similar tasks in the future?	yes, most likely	no, not at all
2:6	How satisfied are you with your proposal of a solution for the task?	unsatisfied	satisfied
2:7	Did the program disturb you in your work on trying to solve the task?	no, not at all	yes, very much
2:8	Did the program support you in your work on trying to solve the task?	no, not at all	yes, very much
2:9	Was the “behaviour” of the program consistent?	no	yes

Table B.1 General questions concerning the design environment
(translated from Swedish)

B.2 Questions about the commenting tool (general)

#	Question	Scale endpoints (1–8)	
3:1	Did you consider the comments from the commenting system (CS) to be:	not useful	useful

Table B.2 General questions concerning the commenting tool
(translated from Swedish)

#	Question	Scale endpoints (1–8)	
3:2	How did you perceive the speed of the CS?	too slow	too fast
3:3	How did you perceive the comments from the CS?	easy to comprehend	hard to comprehend
3:4	How did you perceive the competence of the CS?	insufficient	sufficient
3:5	If you were given a similar task in the future – would you then like to use a system that delivered comments?	no	yes
3:6	Did the design-part of the system hinder you to solve the task?	no, not at all	yes, very much
3:7	How would you describe the comments delivered by the system?	disturbing	supporting
3:8	Was it easy to locate the objects referred to in the comments?	no	yes

Table B.2 General questions concerning the commenting tool
(translated from Swedish)

B.3 Questions about a specific situation

#	Question	Scale endpoints (1–8)	
c:1	Did you consider the comments from the commenting system (CS) to be:	not useful	useful
c:2	How much of the comment did you read?	none of it	all of it
c:3	How did you perceive the comment from the CS?	easy to comprehend	hard to comprehend
c:4	Were you already familiar with the contents of the comment?	no, none of it	yes, all of it

Table B.3 Situation-specific questions (translated from Swedish)

#	Question	Scale endpoints (1–8)	
c:5	How did you perceive the speed of the CS?	too slow	too fast
c:6	How would you describe the comments delivered by the system?	disturbing	supporting
c:7	Was it easy to locate the objects referred to in the comments?	no	yes

Table B.3 Situation-specific questions (translated from Swedish)

Appendix C

Perceived MWL

Measurement Forms

C.1 TLX form

For each pair in the table below, mark the factor that you perceived as causing the greatest workload for you.

<input type="checkbox"/> Physical demand/ <input type="checkbox"/> Mental demand	<input type="checkbox"/> Temporal demand/ <input type="checkbox"/> Physical demand	<input type="checkbox"/> Effort/ <input type="checkbox"/> Performance
<input type="checkbox"/> Temporal demand/ <input type="checkbox"/> Mental demand	<input type="checkbox"/> Performance/ <input type="checkbox"/> Physical demand	<input type="checkbox"/> Frustration/ <input type="checkbox"/> Performance
<input type="checkbox"/> Performance/ <input type="checkbox"/> Mental demand	<input type="checkbox"/> Effort/ <input type="checkbox"/> Physical demand	<input type="checkbox"/> Effort/ <input type="checkbox"/> Temporal demand
<input type="checkbox"/> Effort/ <input type="checkbox"/> Mental demand	<input type="checkbox"/> Frustration/ <input type="checkbox"/> Physical demand	<input type="checkbox"/> Frustration/ <input type="checkbox"/> Temporal demand
<input type="checkbox"/> Frustration/ <input type="checkbox"/> Mental demand	<input type="checkbox"/> Temporal demand/ <input type="checkbox"/> Performance	<input type="checkbox"/> Effort/ <input type="checkbox"/> Frustration

For each scale, describe how you perceived the size of each factor by putting a cross on the line.

Physical demand	<i>low</i>	_____	<i>high</i>
Temporal demand	<i>low</i>	_____	<i>high</i>
Performance	<i>bad</i>	_____	<i>good</i>
Effort	<i>low</i>	_____	<i>high</i>
Mental demand	<i>low</i>	_____	<i>high</i>
Frustration	<i>low</i>	_____	<i>high</i>

Figure 8.2 The TLX form used to measure perceived MWL (translated from Swedish). Note that RTLX was used for the measurement, i.e., the upper half of the form was not analysed.

Appendix D

Evaluation-task material

D.1 The experts' written instructions

The following text (in Swedish) was used to instruct the experts during the evaluation of the interface proposals (see sections 3.5 and 5.5).

Evaluation of CODEK interface design proposals

In front of you, you have print-outs of three different design proposals for the interface of an hotel reservation application. You also have the “specification” used by the designers of the proposals.

Your task is to evaluate the three proposals, and to deliver comments/critique concerning them.

Procedure

For each comment/critique – mark the interaction object on the print-out (using a number or in a similar manner), and use a separate piece of paper to describe what you want to say.

If the comment doesn't refer to any specific interaction object, explain which (and which part) of the three proposals it refers to.

You may evaluate the proposals one at the time or in parallel.

You are not allowed to use more than a total of 30 minutes for the evaluation.

Remember

During the evaluation, you should bear in mind that the proposals are designed in a tool that:

- can only be used to design the static parts of an interface, i.e. the graphical layout
- only provides the interaction objects depicted on the next page [see figure 4.4]

You should only deliver what could be interpreted as useful/usable comments/critique, i.e., what you say should contribute positively to the usability of the redesigned application.

About the proposals

Each proposal contains one or more windows, which are shown as separate images. In proposal 2 and 3 there are images with the letters “A” and “B” below – those images should be interpreted as follows:

- They contain interaction objects with several “levels”, e.g., option menus or pull-down menus.
- “A” is the image normally seen by the user.
- “B” shows the multi-level object “extended”.

D.2 The interface proposals



Figure D.1 Interface proposal 1



Figure D.2 Interface proposal 2

A

SSK hotel

Ticket period: 1/1 - 1/1

Hotel room: 1, 2, 3, 4

Book actions

B



A

B

Hotel	Four hotel	
Address	Kingston 6A 123 45 Street	
Telephone	823-456789	My address <input type="button" value="Delete"/>
Account	Versicherungsges. Büro GmbH	<input type="button" value="Kingston"/> <input type="button" value="Harris"/>

Resultat

Inga hotell finns.

[Slippa detta, ta i hand](#) [My arkiv](#)

Figure D.3 Interface proposal 3

Appendix E

Session interaction diagrams

E.1 Contents of Interaction Diagrams

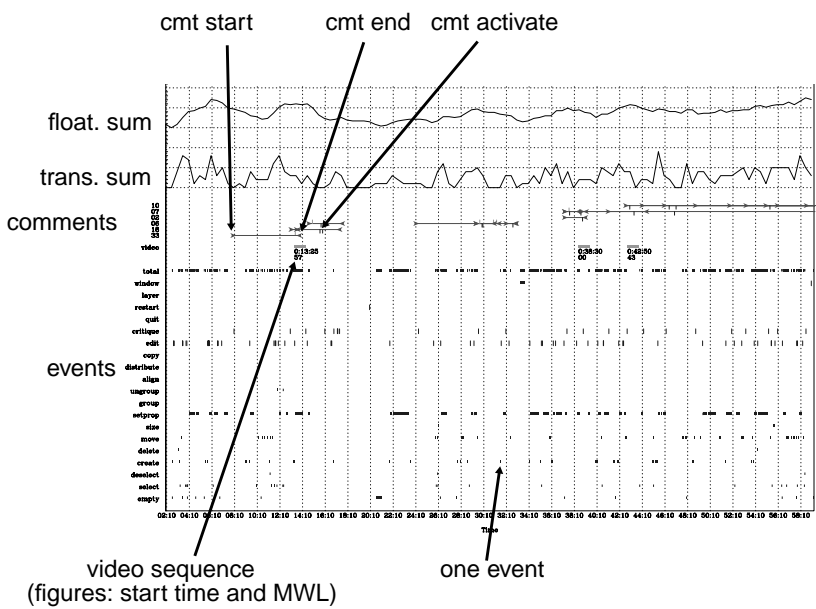


Figure 8.3 Example of contents in an Interaction Diagram

E.2 Interaction Diagrams 9, 14 and 16

The interaction diagrams 9, 14 and 16 are described and discussed in section 6.7 and section 7.5.

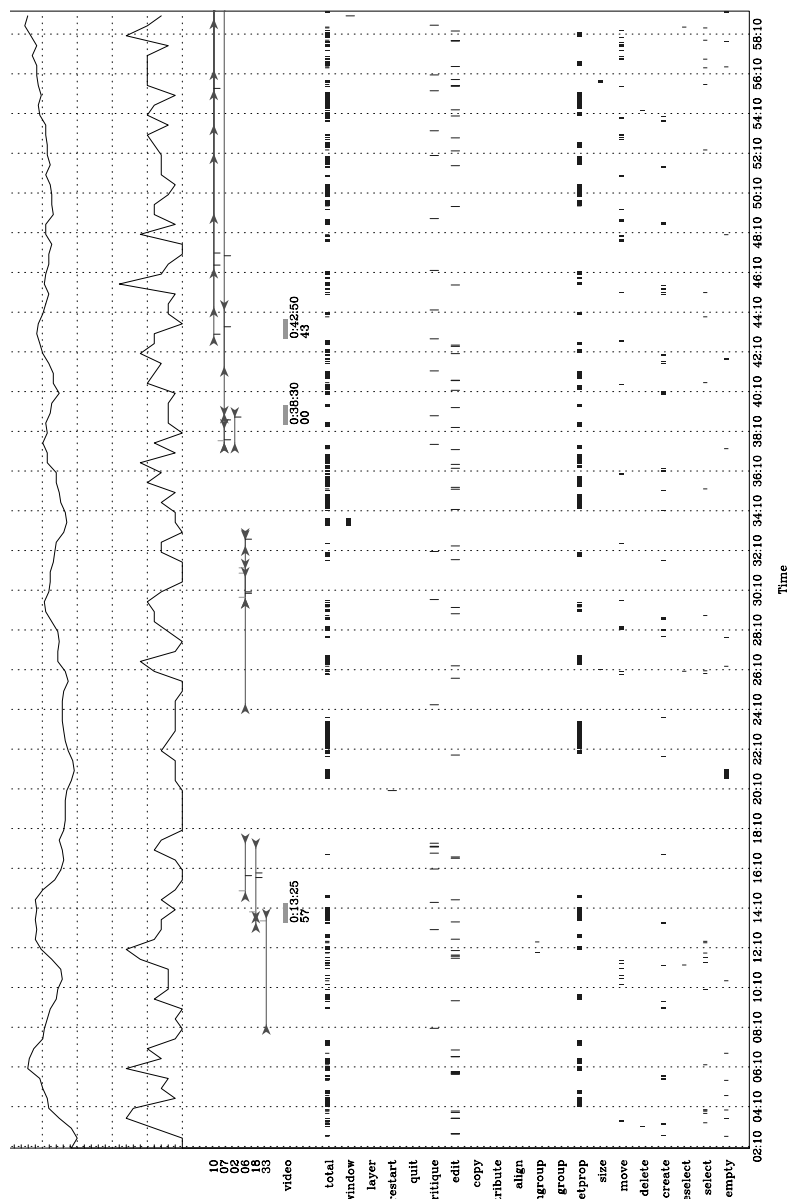
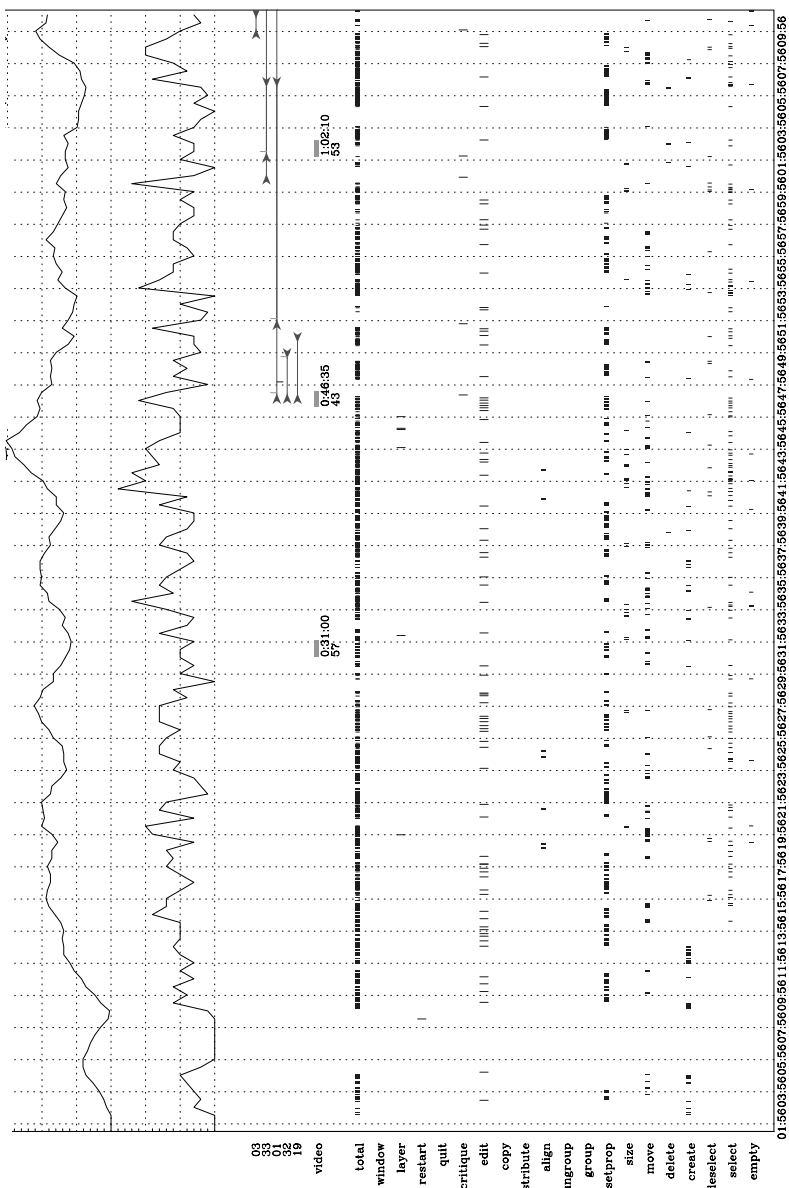


Figure E.1 ID 9



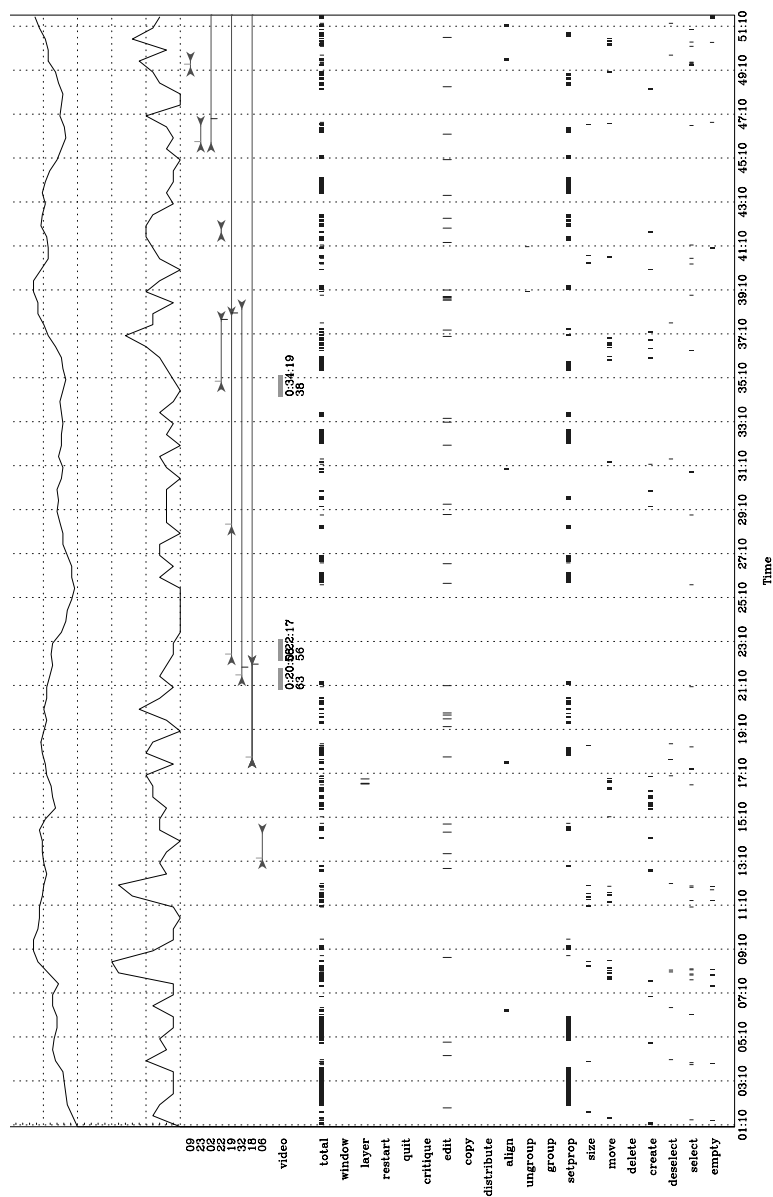


Figure E.3 ID 16



Avdelning, Institution
Division, department

Department of Computer and
Information Science

Institutionen för datavetenskap

Datum
Date

October 1996

Språk

Language

- ☐ Svenska/Swedish
☒ Engelska/English

☐ _____

Rapporttyp

Report: category

- ☒ Licentiatavhandling
☐ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport

☐ _____

ISBN

91-7871-833-3

ISRN

LiU-Tek-Lic 1996:41

Serietitel och serienummer

Title of series, numbering

ISSN

0280-7971

Linköping Studies in Science and Technology

Thesis No 576

URL för elektronisk version

Titel

Title

Commenting Systems as Design Support—A Wizard-of-Oz Study

Författare

Author

Mikael Ericsson

Sammanfattning

Abstract

User-interface design is an activity with high knowledge requirements, as evidenced by scientific studies, professional practice, and the amounts of textbooks and courses on the subject. A concrete example of the professional need for design knowledge is the increasing tendency of customers in industrial systems development to require style-guide compliance. The use of knowledge-based tools, capable of generating comments on an evolving design, is seen as a promising approach to providing user-interface designers with some of the knowledge they need in their work. However, there is a lack of empirical explorations of the idea.

We have conducted a Wizard-of-Oz study in which the usefulness of a commenting tool integrated in a design environment was investigated. The usefulness measure was based on the user's perception of the tool. In addition, the appropriateness of different commenting strategies was studied: presentation form (declarative or imperative) and delivery timing (active or passive).

The results show that a commenting tool is seen as disturbing but useful. Comparisons of different strategies show that comments from an active tool risk being overlooked, and that comments pointing out ways of overcoming identified design problems are the easiest to understand. The results and conclusions are used to provide guidance for future research as well as tool development.

Nyckelord

Keywords

User Interface Design Support, Commenting, Critiquing, User Interface Design Environment, Wizard-of-Oz, Empirical Study, Agents