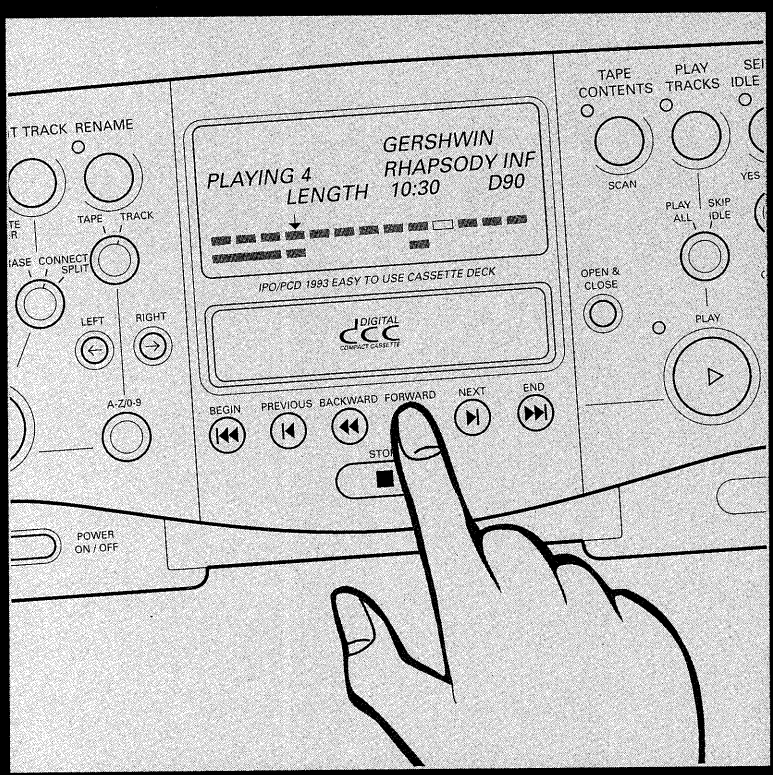
Layered feedback in user-system interaction

Layered Feedback in User-System Interaction

Reinder Haakma



Layered Feedback in User-System Interaction

Haakma, Reinder

Layered Feedback in User-System Interaction /
Reinder Haakma
Thesis, Eindhoven University of Technology
ISBN 90-74445-40-3
Subject headings: layered feedback / man-machine interaction / user interfaces

Cover by Peter Doodson and Marjolein de Vette.

The work described in this thesis was carried out at the Center for Research on User-System Interaction (IPO) as part of the Philips Research programme.

Layered Feedback in User-System Interaction

proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Eindhoven
op gezag van de Rector Magnificus, prof. dr. M. Rem,
voor een commissie aangewezen door het College voor Promoties
in het openbaar te verdedigen op
dinsdag 30 juni 1998 om 14.00 uur

door

Reinder Haakma

geboren te Leeuwarden.

Dit proefschrift is goedgekeurd door de promotoren:

prof. dr. D.G. Bouwhuis
en
prof. dr. A. Vandierendonck

Acknowledgments

I thank my colleagues at IPO and Philips for their help and support and my family for their encouragement.

Table of contents

Chapter 1

Introduction

The central theme of this thesis is how to design easy-to-use user interfaces for novice and occasional users.

1.1 Design approaches

There are different approaches to user-interface design. *Traditionally*, user interfaces of new products are modified versions of those of older products, leading to highly standardized, conventional user interfaces. This approach is dominant in the consumer electronics industry. The user interfaces providing access to the core functionality of, for example, cassette recorders and CD players, are highly similar.

This traditional approach contrasts with the *design approach (see e.g. Kristof & Satran, 1995)*. The idea is that good user interfaces are designed by gifted artists. User interface design is viewed as a highly creative process. The value of this approach is that it may lead to new interaction concepts. User interfaces created this way, are often unconventional and innovative. In general, the aesthetic quality is high. The usability of the resulting user interface depends very much on the talent and experience of the individual artist.

Another approach is the *iterative design* procedure (see e.g. Hix & Hartson, 1993). User interfaces are designed, evaluated and re-designed until a sufficient level of usability has been reached. The good thing about this approach is that the usability of the user interface is actually assessed. The quality of the resulting user interface is less dependent on the skills of the designers. The design *process* guarantees a sufficient level of usability.
In general, the number of design iterations is limited. This means that the resulting user interface is to a great extent determined by the initial user interface. On basis of the test results, the initial user interface is modified until the most serious usability problems have been resolved.

A fourth approach is the *model-based* user-interface design. User interfaces are designed according to a general framework describing how to structure user-system interaction to arrive at good usability and indicating how design decisions influence usability. The advantage of this approach is that an integrated body of knowledge about user interfaces and their usability is used in the design process. The outcome of the process is less dependent on the individual talents of the designers because of the guidance provided by the framework. The main problem is that there is no such framework for user interfaces to be used by novice and occasional users.

1.2 Model-based user-interface design

The GOMS model (Card, Moran and Newell, 1983) is the best-known model guiding the design of user interfaces to be used by expert users (John, 1990). By analyzing the user-system interaction as it should evolve to successfully carry out a set of representative tasks, an inventory can be made of the physical and cognitive processes experts have to execute. By estimating the individual process-execution times, the overall task execution times can be computed and thus the efficiency with which experts can fulfil their tasks. To arrive at optimum user productivity, the most efficient user interface must be selected from a set of alternative user interfaces. The users have to be experts because the GOMS model assumes that users have internalized the interaction procedures and do not spend time on learning them: the interaction is assumed to unfold according to a protocol mutually understood by the user and the system.
Extensions to the GOMS model have been developed to incorporate learning processes, for example CTT (Cognitive Complexity Theory, Kieras and Polson, 1985) and TAG (Task-Action Grammars, Payne and Green, 1986). These two models define a complexity metric that indicates how hard it is for users to learn the interaction rules. However, the complexity of a user interface is calculated only on the basis of a rule-based description of the task-execution procedures. These models do not take into account what information users need for the selection and execution of the interaction procedures, nor to what extent this information is provided by the system in the form of feedback.

The model that takes both task-execution procedure and feedback into account is the Layered Protocols (LP) framework (Taylor, 1988a, 1988b). LP comprises the procedural aspects of the interaction by introducing the General Protocol Grammar (GPG). Central in the LP framework is that a hierarchical task-action decomposition requires a hierarchically organized interaction with feedback provided at each level. In this way, LP tightly links the procedural and the feedback aspects of user-system interaction. This thesis focuses on the LP framework because important factors influencing the speed at which novice users learn to operate a system are likely to originate from this link.

1.3 Overview of the thesis

Because of the focus on the LP framework, the central theme of this thesis was changed to how the LP framework can provide guidance in the design of easy-to-learn user interfaces for novice and occasional users. This focus was not without problems. The LP framework described by Taylor in his two 1988 papers was rather informal. Taylor had introduced many potentially useful concepts, but which are the most important ones? Also, no attempts had ever been made to use the LP framework to design full-scale user interfaces. Can this be done? And, last but not least, does the LP framework guide UI designers towards more usable interfaces?
So, together with my colleagues, I started working towards a more clearly defined framework that can be used for designing full-scale user interfaces and that provides guidance toward easy-to-learn user interfaces for novice and occasional users.

In a first step, together with my colleague Frits Engel, I carefully studied the concepts introduced by the Layered Protocols framework. We elaborated upon the role of expectations in user-system interaction. This study resulted in a first publication called 'Efficiency in man-machine communication' in the 1990 Annual Progress Report of IPO (Haakma & Engel, 1990). Further study and discussions led to a publication in the Philips Journal of Research called 'Layered Approach of user-system interaction' (Engel & Haakma, 1992). Also, we submitted an expanded version of this paper for publication in the International Journal of Man-Machine Studies. A modified version of that paper was accepted for publication in 1993. This paper, called 'Expectations and feedback in user-system communication' (Engel & Haakma, 1993), is chapter 2 of this thesis. These three publications aimed at obtaining a good understanding of the LP framework, determining the framework's core concepts and making those concepts operational in user-interface design.

In the meantime, I became involved in the DRUID project, a Philips Research/IPO project. DRUID is the acronym of Digital-Recorder User-Interface Development. The project's aim was to develop innovative user interfaces for consumer equipment. More in particular, the goal was to develop an 'easy-to-use' user interface for Digital Compact Cassette (DCC) recorders. The project created a unique opportunity to put the LP framework to the test. Would it be possible to design a full-scale user interface based on the LP framework, and what impact would LP have on the product's usability?
Together with my colleagues Berry Eggen and Joyce Westerink, I designed a user interface for DCC recorders using the LP framework. The design method was a combination of the model-based approach and the design approach. The design approach dominated, but the interaction was hierarchically structured as indicated by the LP framework.
Next, the newly designed DCC recorder was tested with users. Its usability was compared with that of a commercially available DAT (Digital Audio Tape) recorder. Offering largely the same functionality, the DCC recorder turned out to be more usable than the DAT recorder, due mostly to factors that could be explained in LP concepts. This work resulted in a paper called 'Layered Protocols: Hands-on experience' that was published in the International Journal for Human-Computer Studies in 1996. This paper is chapter 3 of this thesis.

Together, the two publications in IJHCS describe a first evaluation round of the LP framework: the framework was studied and made operational, a full-scale user interface was designed according to the framework, the interface was tested with users, its usability was benchmarked, and the evaluation results were linked back to the LP framework. The second part of this thesis describes another, more detailed evaluation round of the LP framework.

During the experiment, Berry, Joyce and I were often puzzled by the observed behavior. Were there any psychological models that could help us better understand why these novice users behaved as observed in the evaluations? The first idea was that novice users, not knowing the interaction rules, would use problem solving-strategies to achieve their intentions (Newell and Simon, 1972, VanLehn, 1989) and, in the meantime, learn the interaction rules. In their 1990 paper, Polson and Lewis introduced such a model of user

learning. Independently of this, Taylor (1992) had developed the LP framework further and linked it to Perceptual Control Theory (Powers, 1973). Perceptual Control Theory (PCT) states that behavior is the control of perception. This would mean that the behavior of novice users should be explained in terms of user perceptions created by the system. In my paper called 'Why do novice users do what they do' it is illustrated how behavior of novice users can be explained by perceptions created by the system that trigger some basic problem-solving strategies in users. This paper has been submitted for publication in the IJHCS. Chapter 4 of this thesis, called 'Towards explaining the behavior of novice users', is a slightly modified version of this paper.

In a next step, together with my colleague René Wijnen, I created another, very different user interface for DCC recorders. The user interface was developed in a systematic, hierarchical way. Its description, presented in chapter 5, reflects this. Each interaction level was designed independently of the underlying layers. The interaction at the various levels was described using the same format.
In this way we tested whether it was feasible to describe a user-interface in a completely systematic way and to what extent interaction protocols can be designed independently of each other. In the descriptions of the various interaction protocols, we described not only their internal organization but also the feedback which, according to the Perceptual Control Theory, is required for successful user-system interaction.

Though we had now shown that model-based user-interface design is feasible using the LP framework, we still had to find out whether our increased understanding had led to a more usable recorder. That is why René Wijnen and I tested the second DCC user interface in the same way as the other two interfaces had been evaluated. This evaluation is described in chapter 6. The user interface proved to be more usable than both the DAT recorder and the first DCC recorder. The points at which the interaction between user and system was not as fluent as expected could be explained in terms of the LP framework.

Chapter 6 closes the second evaluation round testing whether full-scale model-based user-interface design using the Layered Protocol framework is feasible and whether the framework can guide designers towards user interfaces with higher usability for novice and occasional users.

The concluding chapter of this thesis can be seen as the start of a third evaluation round. It discusses the main results of the work presented in this thesis and provides a road map for further development of the LP framework towards a proven and practical model guiding user-interface design.

This thesis hence provides an outline of a concise and operational version of the Layered Protocols framework, it illustrates how full-scale user interfaces can be designed according to the framework, and it shows that structuring user-system interaction according to the Layered Protocols framework can lead to more usable interfaces for novice and occasional users. I hope that this thesis may contribute towards a design approach that is predominantly model-based, that leaves sufficient room for design creativity and leads to highly usable user-interfaces requiring only a minimum number of design iterations.

1.4 References

Card, S.K., Moran, T.P. & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Eggen, J.H., Haakma, R. & Westerink, J.H.D.M. (1996). Layered Protocols: hands-on experience. *International Journal of Human-Computer Studies,* **44**, 45-72.

Engel, F.L. & Haakma, R. (1992). Layered approach in user-system interaction. *Philips Journal of Research,* **47**, 63-80.

Engel, F.L. & Haakma, R. (1993). Expectations and feedback in user-system communication. *International Journal of Man-Machine Studies,* **39**, 427-452.

Haakma, R. & Engel, F.L. (1990). Efficiency in man-machine communication. *IPO Annual Progress Report*, **25**, 90-96.

Hix, D. & Hartson, H.R. (1993). *Developing User Interfaces*. New York: Wiley.

John, B.E.(1990). Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. In: *Proceedings of CHI, 1990*. NewYork, ACM.

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

Kristof, R. & Satran, A. (1995). *Interactivity by Design. Mountain View.* CA: Adobe Press.

Newell, A. and Simon, H.A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

Payne, S. J. & Green, T. R. G. (1986). Task-action grammars: a model of the mental representation of task languages. *Human-Computer Interaction*, **2**, 93-133.

Polson, P.G. & Lewis, C.H. (1990). Theory Based Design for Easily Learned Interfaces. *Human-Computer Interaction*, **5**, 191-220.

Powers, W.T. (1973). *Behavior: the control of perception*. Chicago, Illinois: Aldine Publishing Company.

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

Taylor, M.M. (1988b). Layered protocols for computer-human dialogue. II: Some practical issues, *International Journal of Man-Machine Studies*, **28**, pp. 219-257.

Taylor, M.M. (1992). *Principles for Intelligent Human-Computer Interaction, a tutorial on Layered Protocol Theory*. North York, Ontario: DCIEM report no. 93-32.

VanLehn, K. (1989). Problem Solving and Cognitive Skill Acquisition. In: Posner, M.I. (Ed.), *Foundations of Cognitive Science*. London: MIT Press.

Chapter 2

Expectations and feedback in user-system communication[1]

Abstract

In terms of speed and accuracy of intention transfer, normal human conversation proves to be very efficient: exchanged messages only carry sufficient information relative to contextual knowledge assumed to be present at the receiver's end. Furthermore, by receiving layered feedback from the recipient, the speaker is able to verify at an early stage of communication whether his intentions are being accurately perceived. Finally, in divergencies from the expected messages, the listener may ask for clarification at an early stage of message interpretation.

For user-system communication to become similarly more efficient, machine interfaces should display both early layered (I-)feedback about partial message interpretations as well as layered expectations (E-feedback) about the message components still to be received. Examples of interfaces are given which already possess these desirable characteristics in part.

The 'layered protocol model', proposed by Taylor (1988a) as a framework for user-system interface design, details the use of layered I-feedback and related repair messages in user-system communication. In this paper we suggest that the model can be improved by providing it with layered E-feedback, as derived from assumed intentions and layered knowledge of the interaction history.

2.1 Introduction

Nowadays, interactive human-computer interfaces are characterized by the use of graphical displays and two-dimensional input devices such as the mouse. A metaphorical display of the domain of application is used often to let the user have initial expectations about the available functionality. Also, developments are in progress to extend the graphical user interface further with multimedia capabilities, such as video, audio and natural language I/O. Of course, this increase in bandwidth and number of channels is relevant

1. This chapter has been published as a paper by Engel, F.L. & Haakma, R. (1993) in the International Journal of Man-Machine Studies, volume 39, pp. 427-452.

for unhampered communication. However, as the functionality of these computer-supported systems continues to increase, the increased complexity of the interaction procedures and the difficulty of mastering them will become prime issues in user interface design. See, for instance, Bösser (1987) for a good survey and further elaboration of learning aspects in human-computer interaction.

To make their control more efficient for the user and also easier to learn, these complex computer-based appliances will need extensive and, in particular, interactive user support (such as interactive error correction, ambiguity resolution, help services, documentation and tutoring facilities). We assume that a considerable part of the increasing communication capacity will be utilized for that purpose in the near future. The main research question then is how to apply the increased communication capacity for a self-evident and efficient method of interactive user support.

In order to illustrate more fully what is involved in efficient and more natural user-system interaction[1], we will first briefly describe the layered protocol model of computer-human dialogue as proposed by Taylor (1988a). This conceptual framework will help us understand the relevance of layered feedback about partial message interpretations (here called I-feedback) for efficient interaction.

Thereafter we will concentrate on the active use of receiver expectations in user-system communication, as another factor of relevance in communication efficiency. Supported by the analysis of some existing user interfaces that already make favourable use of layered feedback and/or machine expectations, the concept of E-feedback will be added to the layered communication model of Taylor (1988a). It is claimed that this extension further assists in the systematic design and analysis of user support for efficient man-machine communication.

2.2 Taylor's layered-protocol model

2.2.1 Intention transfer

To obtain information, to let others (men or machines) know or to have them do things, one has to communicate ones intentions by the exchange of messages. Messages describe these intentions in coded form, their coding being more or less adapted to the transmission characteristics (modulation parameters, bandwidth, error probability, etc.) of the specific communication channels used.

---

1. Note that we assume here without further proof that communication efficiency, as determined by the speed and accuracy of intention transfer from user to system, is a major component of 'user friendliness'. A more natural user-system interaction is better attuned to human communication capabilities and requires less learning because of its consonance with already mastered human conversation protocols.

Accordingly, the sender encodes the intention(s) concerned into messages to be transmitted to the receiving party by modulation of the physical parameters of the selected communication channel(s). In turn, the message receiver tries to decode the incoming messages back again into the original intentions of the sender, but usually doesn't succeed directly in getting the complete original intentions. To handle possible faults (errors and/or ambiguities) in communication of the primary intention, the exchange of secondary messages is therefore frequently needed, see Figure 1.

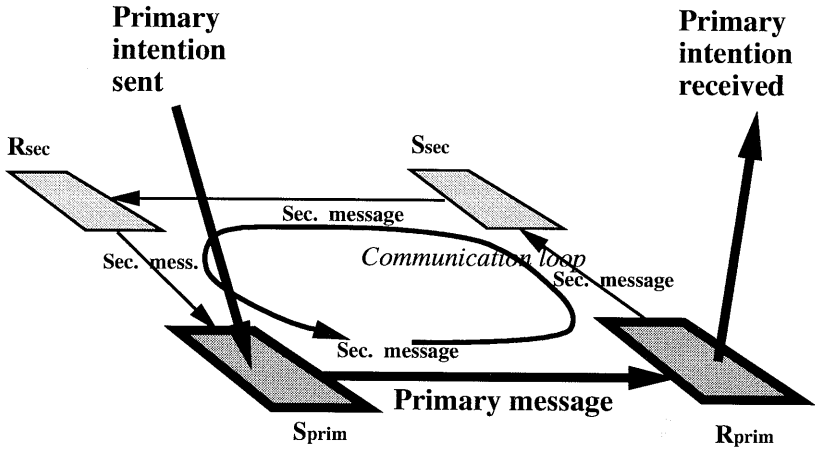


FIGURE 1. Primary intentions have to be transferred from sender Sprim to receiver Rprim by encoding them into primary messages suitable for transfer to the receiver by modulation of the physical parameters of the selected communication channel(s). In turn, the receiver of a message is assumed to decode the incoming messages with the help of the same communication protocol back again into the intentions originally sent. To minimize possible errors and ambiguities in communication of the primary intention, the exchange of secondary messages is frequently needed. For that purpose the communication loop is closed via sender Ssec of the recipient and receiver Rsec of the originator of the primary message.

In communication between machines, the coding and decoding processes are based in general on fixed, context-independent, coding conventions, called communication protocols. Accordingly, errors in communication between machines mainly arise from physical disturbances in the transmission channels used.

In contrast to communication between machines, messages in human communication carry generally just sufficient information in relation to the knowledge context already available, see e.g. Barwise & Perry (1983). Accordingly, the coding and decoding processes in human communication are strongly correlated with the changing knowledge and beliefs (unverified knowledge) of the interacting parties; see, for instance, Beun (1989, pp. 95-120). This context-dependency makes communication considerably faster. For instance,

by simply referring to what has been mentioned earlier in the dialogue, anaphora (see, for instance, van Deemter, 1991) avoids the need for repeated detailed description of what is meant.

Context-dependence also implies, however, that specific errors and/or ambiguities in coding/decoding may arise, resulting from differences in the contextual knowledge applied and assumed by sender and receiver. Mainly by feedback, viz. by observing the reactions of the recipient to the messages sent, it is possible for the sender to detect faults in the decoding.

2.2.2 Layered feedback in human communication

In the human information-processing cycle, which proceeds from sensory perception, via mental processing back to physical action, several stages of coding and decoding can be discerned. To account for differences in reaction time for tasks of different complexity, Donders as long ago as 1868 (quoted by Boring, 1950) suggested successive processing stages in perception. On the production side, where conversion from intention to action takes place, evidence for successive processing stages emerges from, among other things, analysis of action slips (Norman, 1981) and self-repair in speech (Levelt, 1983).

The layered-protocol (LP) model of Taylor (1988a), developed for the description of inter-human communication as well as for the design of human-computer interfaces, assumes that the primary intention to be transmitted is repeatedly broken down into sequences of more detailed temporal combinations of lower-level sub-intentions, until at the lowest level these (sub-)subintentions are converted into elemental physical actions which modulate the parameters of the communication channel(s) involved. The reverse is assumed to happen at the recipient's end, where incoming messages in terms of value fluctuations of the transmission parameters are sensed and converted into sequences of decoded sub-intentions, which are then recognized in turn as representing higher-level (sub-)intentions, etc.

The LP model makes a distinction between the concepts 'message' and 'intention'. Messages are assumed to represent intentions in a communicable form, i.e. they have to be communicated to achieve their intention. Accordingly, the encoders are assumed to transform each incoming higher-level intention message into a sequence of more detailed output messages, containing in their turn the related lower-level sub-intentions. Note that in this approach semantics (intention) and syntax (message structure) exist at each level. The relation with the traditional linguistic distinction between phonemic/alphabetic, lexical, syntactic and semantic levels will be elaborated further in the final discussion.

By dividing the receiver's message-to-intention decoding process into a number of successive processing steps, feedback concerning the intermediate decoding results can be given to the message sender at an early stage, thus enabling the latter to prevent accumulation of faults and in doing so to improve the efficiency (speed versus accuracy trade-off) of communication.

The essence of feedback is that the output of a (communication) process is compared to the corresponding input, so that feedback from a specific message-decoding step at the receiver's end should be addressed to the corresponding encoder step at the sender's end. A consequence of this assumption of mutually corresponding coding and decoding stages, combined with stage-specific feedback, is that layers of communication loops can be distinguished in the interaction between parties; see Figure 2.

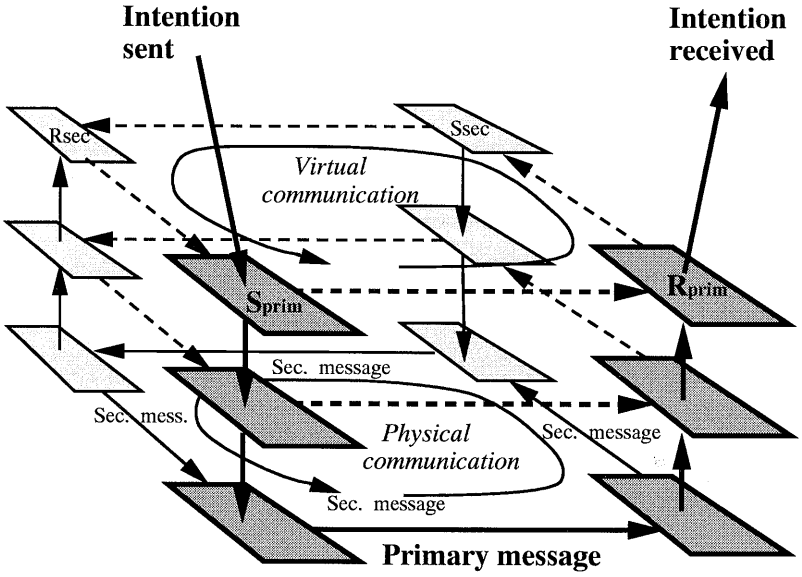


FIGURE 2. Layered communication between mutually related message encoding and decoding stages. On the left, in bold face, the incoming primary intention is progressively encoded by the sender column (Sprim) until, at the lowest layer, the primary message is physically transmitted to the receiver column (Rprim) on the right. Rprim will progressively decode the message back until the original intention is recovered again. Ssec and Rsec, respectively, represent the secondary message sender and receiver column needed for the exchange of feedback messages, calls for clarification of the primary message and other secondary messages. Coding and decoding stages are assumed to correspond reciprocally, enabling the layered (virtual) communication of secondary messages.

In normal human discourse, where several communication channels of different sensory modality are simultaneously available, message elements are frequently transmitted in parallel. Voice is frequently combined with, for instance, gestures, direction of gaze, etc. Given that these multi-modal message elements have to be combined again at some intermediate stage of decoding, their use is consistent with the idea of layered coding in human communication.

Note that all higher-level exchange of intentions between corresponding stages have to be communicated by modulation of transmission parameters at the lowest level. In line with the well-known ISO-OSI reference model for communication between computers (Tanenbaum, 1981), Taylor (1988a) draws a distinction between physical communication at the lowest level and (layer-specific) virtual communication at the higher levels. In the context of human communication, Taylor's model differs from the ISO-OSI model for computer communication in, among other things the explicit use of feedback at each layer. In fact, the distinction of individual layers in his model depends on the possibility of feedback at the given level of abstraction.

A further characteristic of Taylor's LP model is the assumption that message exchange at the different layers is described by the same grammar, the 'General Protocol Grammar'; see Taylor (1988b). It gives the possible sequences of primary and secondary messages of a given layer by means of a state-transition diagram.

2.3 Expectations in human perception

The use of context knowledge significantly enhances the accuracy and speed of inter-human communication. Its influence on human speech perception in noise has been assessed experimentally by, for instance, Miller, Heise & Lichten (1951), who reported that content words (nouns, verbs, adjectives, adverbs) are more intelligible when heard in the context of a grammatical sentence than when scrambled and spoken in isolation as items on a list. In a more subtle demonstration of this effect, Miller & Isard (1963) used sentences with different levels of syntactic and semantic context. Marslen-Wilson (1980) gave further experimental evidence for the positive effect of semantic and syntactic context by determining the minimum length of the word fragment needed for its recognition. She showed that context information provides the constraints needed for eliminating the alternatives from what she called the word initial cohort, the set of words beginning with the same phonemic sequence. Similar effects of context can be found in reading, for an overview see Taylor & Taylor (1983).

Human reading and speech understanding, can be seen as interactive processes in themselves. By what is called analysis-by-synthesis (e.g. Neisser, 1967; Norman, 1969), the data-driven, bottom-up recognition processes are assisted by top-down concept driven planning processes, which support message interpretation by the generation of expectations. These expectations arise at different levels of decoding. For their generation, syntactic (structural) and semantic (interpretative) context knowledge is used at the levels of phonemes, words and utterances and even at the level of discourse. In human discourse, by giving first a rough indication of the primary intention of communication, this contextual knowledge enables the recipient to generate expectations about what will follow. This effect was clearly observed in, for instance, the transcriptions of telephone calls from travel-information seekers (Beun, 1989, pp. 81-93).

Decoding efficiency is expected to improve in a number of ways. First, it is assumed that the look-up activity will be reduced by early selection of the relevant set (lexicon) of decoding transformations. Furthermore, in the case of disagreement with the items in the lexicon, the recipient may ask for elucidation at an early stage in message decoding. Finally, in a situation where a listener is so quick to react that he utters his anticipations even before the speaker has expressed his intentions, the speaker only needs to confirm (one of) them, thus again effecting an increase in speed.

Feedback, requests for clarification in the case of unsatisfied expectations, and also certain repair actions in the case of detected errors and/or ambiguities in communication, constitute the secondary messages referred to earlier. These secondary messages constitute the "meta-communication" that Oberquelle, Kupka & Maass (1983) found to be missing in computers of the so-called first generation. Bunt, Leopold, Muller and van Katwijk (1978) and, more explicitly, Bunt (1989) described communication which has to do with verification and acknowledgement as "dialogue control acts". Whether this concept also includes repair actions is not clear from their papers. Nevertheless, as Bunt (1988) found for 'information dialogues', for instance, it appears to be quite characteristic of utterances in inter-human communication that more than half of them are secondary messages.

2.4 Machine expectations

In this section examples of human-machine interaction will be analysed and modelled, in which explicit, context-dependent machine expectations are applied. Note that, in general, machine functions already contain certain implicit, context-independent expectations, built into them by the designer/programmer (cf. Oberquelle et al., 1983).

To concentrate our thoughts, we first introduce a simple single-layer, context-sensitive, message-decoder model, containing a message recognizer combined with what will be called an anticipator. This decoder structure will be made plausible by considering a number of applications in which context-dependent machine expectations are incorporated. Thereafter, the multi-layered context-sensitive message decoder model will be introduced, which utilizes level-specific expectations in its communication with the user.

2.4.1 Context-sensitive decoding

The main activity of the message recognizer in a decoder is detection of specific chunks of incoming message elements and assigning interpretations (semantics) to them. To give the ideas a concrete form, it will be assumed that the message recognizer contains different sets (lexicons) of input/output relations. Message expectations are assumed to limit the recognition process by preselecting the most appropriate lexicon needed for the current message-decoding step. The collection of all entries in the selected lexicon then corresponds to the set of expected message chunks at the decoder input.

Although there is hardly a limit to the context knowledge that can be applied for message anticipation, the decoder model is limited for practical reasons to the use of contextual knowledge about the state and history of interaction, combined with knowledge of the global intention of the message to be received. This first approach to a single layer context-sensitive decoder is illustrated in Figure 3.

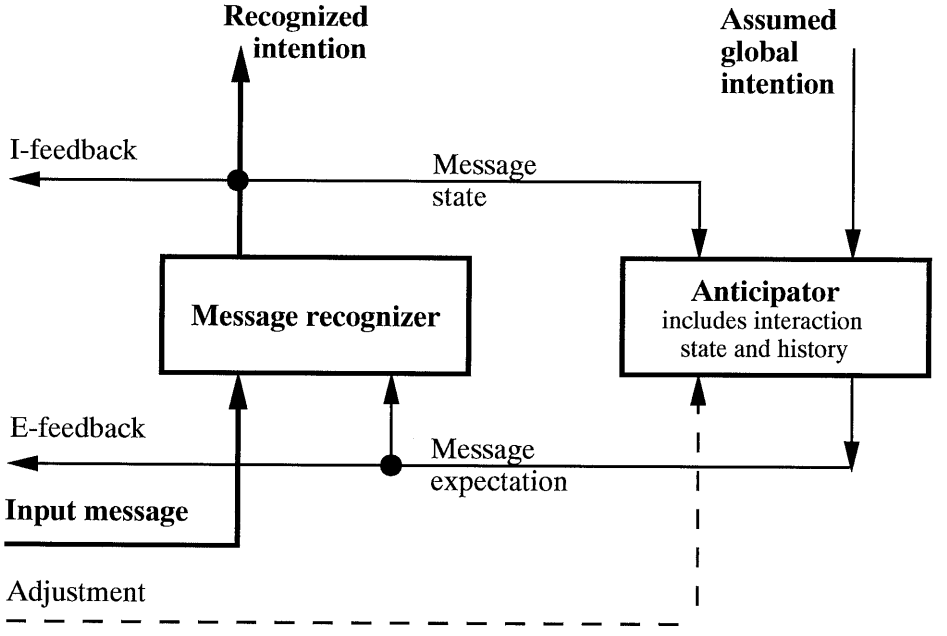


FIGURE 3. Illustration of the main signals and functions distinguished in a context-sensitive message decoder. The conversion from specific chunks of lower level input messages into a recognized (higher-level) intention is made by the message recognizer. The user is assumed to be provided with feedback about already decoded sub-intention(s) (I-feedback). Message expectations constrain the recognition process. The anticipator derives these expectations from knowledge already available about the global intention of the interaction and knowledge of the current state and past history of interaction. In the case of incorrect machine expectations and related E-feedback, the user should be enabled to adjust them.

Figure 3 shows a message recognizer expanded to include an anticipator, together with the relevant signals. The feedback signals intended for the user as the message originator are also shown, namely I-feedback on the message part(s) already interpreted and E-feedback giving the current receiver expectations about the message segment(s) still to come.

Note that strictly speaking, E-feedback is a feed-forward message. We feel this rather loose use of the term feedback to be justified by the fact that in practice I-feedback and E-feedback are frequently combined into a single message. Even from E-feedback alone, certain conclusions may be derived about the status of the recipient's interpretation process.

A clear example of E-feedback in existing user interfaces is in the use of menus. Menus represent context-dependent machine expectations and their use is more or less analogous to the situation in inter-human dialogue, where the recipient states his message anticipations even before the sender has expressed his intentions.

Because machine expectations have in practice to be based on a very limited amount of contextual knowledge and may therefore be erroneous, the display of the related E-feedback is important. It reveals the system's competence and accordingly increases its 'transparency' (Maass, 1983). In cases where these machine expectations are incompatible with the user's intentions, they should be made adjustable. Note that this adjustment may involve more than just a straight correction of that expectation. For instance, also the assumed global intention might be incorrect. This point will be considered in our later discussion of fault handling. For that reason the possibility of adjustment has been provisionally indicated in Figure 3 by a dashed line.

2.4.2 Radio applications

If we examine the interface of a standard radio, it will be clear that a number of simple context-sensitive machine expectations have already been incorporated. For instance, the tuning dial provides not only I-feedback about the selected tuner frequency, but also E-feedback about the tuner frequency expected to be chosen the next time the set is switched on. Likewise, the physical separation of the volume control and the main switch, which were combined in earlier radio types, introduces a built-in context-dependent machine anticipation of this kind, the expectation being that the same volume level will probably be desired when the user switches on again. A direct E-feedback is given in this case by the setting of the volume knob. Note that, with the introduction of digital tuning and digital volume control, designers had to introduce these default values consciously. By requiring less tuning and volume-setting actions, these built-in machine expectations make the interaction clearly more efficient than in the situation where these variables have to be specified each time anew.

Preselection keys represent an important improvement of the car-radio interface. In view of the high probability of interference in radio reception while driving, the built-in assumption as to the intention of interaction is that the driver aims at receiving high-quality sound, and therefore tends to tune to powerful radio transmitters. The preselection keys represent the machine expectation about the user's preferences. They permit a simpler, more efficient manner of interaction, with less visual load and no fine tuning. In fact, a

limited set of key-press to tuner-frequency transformations, the decoding lexicon, is selected by this machine expectation. If necessary, these transformations can be manually adjusted by the user.

Autostore, available in the top-of-the-range car radios from Philips, represents a further advance in radio tuning (see van Nes & van Itegem (1990) for an evaluation of its user interface). It solves the problem of drivers who take their car beyond the transmission range of local FM radio transmitters. A built-in 'autostore procedure' that searches for the five strongest transmitter signals at the current car location and stores their frequencies under the appropriate preselection keys, means that even the interaction needed for adjustment of the machine expectations is simplified. Figure 4 shows a survey of the relationship between the entities in Figure 3 and those of the autostore facility.

Preselection keys with autostore.

Message decoder function: Selection of desired tuner frequency

Message recognizer function: Conversion of tuner control to tuner frequency
Input message:
Full: Activation of tuner control knob
Simplified: Activation of tuner pre-select key
Recognized intention: Desired tuner frequency
I-feedback:
Directly: Selected tuner frequency
Indirectly: Desired high-quality radio sound
E-feedback: Missing, but see 'Extra' below

Anticipator function: Finding of frequencies of 5 locally strongest transmitters
Global intention: High-quality music/info
Expectation: Frequencies of the 5 strongest transmitter signals at the given location
Adjustment: Triggering of the autostore frequency-search procedure
Extra: Bleep signal after successful search for a strong transmitter signal

FIGURE 4. Description of the correspondence between the context-sensitive message-decoding structure given in Figure 3 and a car-radio interface with preselection keys and autostore facility.

2.4.3 Multi-buffer teletext application

Teletext television receivers offer the facility to display pages of textual and/or graphical information, transmitted during the raster-blanking intervals of the TV broadcast signal, instead of the television programme concerned. To minimize electronic page storage in the teletext receivers, the teletext information is broadcast cyclically. The teletext decoder scans the TV signal for the desired page number, after which the relevant page information is loaded into a screen-image buffer and then displayed on the screen.

As the raster-blanking intervals carry only a limited amount of data per unit of time, the page-cycle duration depends on the number of pages. In practice it is about 20 s. As a result, the average time between page-number specification and page loading is also quite large.

To compensate for this latency, advanced Full Level One Features (FLOF) teletext decoders contain more display buffers. The question is then which pages should be loaded in such a multi-buffer Teletext receiver for optimal performance. Depending on the current page on the screen, machine expectations about the most probable candidate page number(s) to be selected next are linked to it. For that purpose, use is made of the informational relations that exist, for instance, between successive menu pages or between parts of text extending over more than a single page. By loading these pages during the reading/display period of the current page, presentation delays are minimized, provided that the desired page is among those expected. Here, too, context-sensitive machine expectations are successfully applied to decrease the page access time and accordingly to increase the efficiency of intention transfer. Figure 5 surveys more closely the relations between the teletext image-buffering strategy and the context-sensitive decoding structure given in Figure 3.

Multi-buffer Teletext

Message decoder function: Selection of desired page number

Message recognizer function: Conversion of key presses to page number
- *Input message:*
  - Full: Keyed-in digits of page number
  - Simplified: Next/previous-page or menu-item number
- *Recognized intention:* Desired page number
- *I-feedback:*
  - Directly: Selected page number
  - Indirectly: Desired page display
- *E-feedback:* None

Anticipator function: Generation of most probable page numbers
- *Global intention:* Fast page access
- *Expectation:* Set of most probable next-page numbers
- *Adjustment:* Not available

FIGURE 5. Survey of the correspondence between Teletext image buffering and the context-sensitive message-decoding structure given in Figure 3.

2.5 Expectations in layered decoding

Now that we have illustrated how machine expectations improve the efficiency of human-machine communication, the next question to be answered is how machine expectations combine with layered communication. In this chapter we consider more closely how layer-specific receiver expectations arise from message history and assumed message intention.

2.5.1 Expectations and the LP model

In conformity with the idea of analysis by synthesis in human message recognition, the LP model can be provided with layered expectations about the message elements still to come. In relation to layered voice recognizers, Taylor (1987) has already hinted that "inter-layer feedback" might determine the momentary set of permitted words as well as their probability of occurrence. The following can be seen as a further elaboration of that idea.

As it is the regular function of each message encoder (planner) to convert incoming higher-level intentions into lower-level communication goals to be subsequently achieved, an assumed message intention, combined with the current state of the sub-message reception/interpretation process, might trigger what is called here an anticipator, to hypothesize about the sub-messages/intentions still to be received. By sending these anticipated sub-messages internally as expectations to the same layer-message recognizer (see Figure 6) the decoding process is ultimately reduced to simple verification of incoming messages.

Of course, the early top-level assumption about the message intention is crucial here. For that reason messages should be structured in such a way that their initial part contains information about the global intention of what follows. From conversational analysis of information dialogues (see, for example, Schegloff, 1980), it is known that in general high-level interaction does indeed start with the supply of information about the topic and further context needed for correct interpretation of what follows. As indicated earlier, a similar effect can be observed in word perception, where the word initial cohort (Marslen-Wilson, 1980) constrains the set of possible characters to follow.

2.5.2 Layered feedback in telephone-number selection

As an example of the model in Figure 6, relating to the use of level-specific expectations and feedback in human-machine communication, telephone-number selection will now be considered. Note that the example is not intended to describe the precise technical realization of this functionality in a telephone exchange.

If one wishes to make a telephone call, a specific interaction with the telephone exchange has first to take place. Figure 7 shows that part of the intention-to-action encoding scheme needed if one wishes to make a national long-distance connection to the Institute for Per-

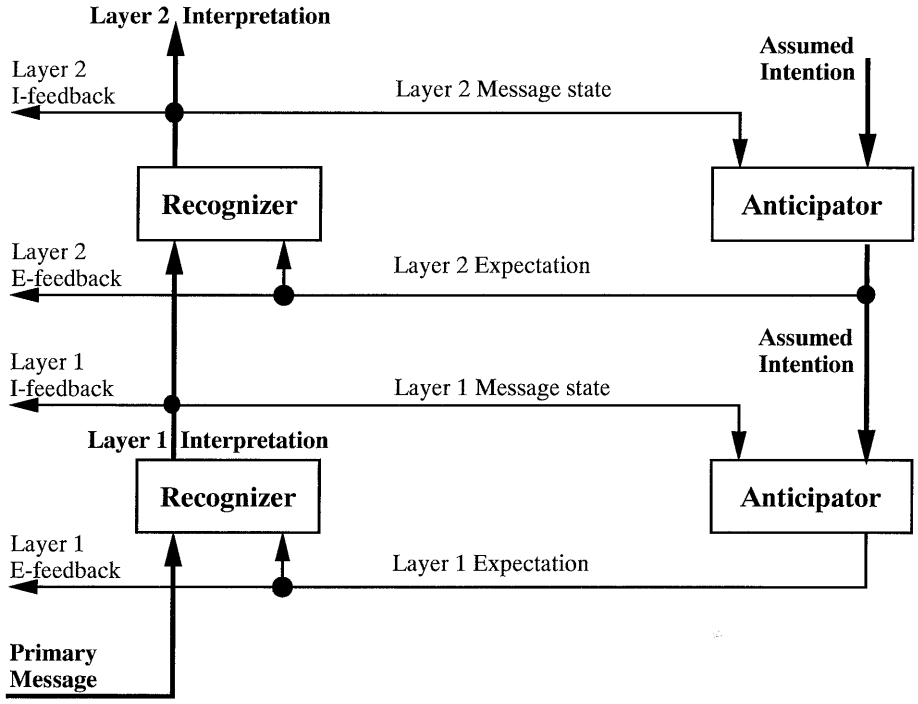


FIGURE 6. Illustration of the extended LP model for human-machine communication. The message anticipators actively support message decoding by generating layered expectations about aspects of the incoming message. These layered expectations are derived from the given state of interaction and the assumed higher-level intention of the incoming message. During message decoding layered I- and E- feedback becomes available to the message originator.

ception Research/IPO in Eindhoven, the Netherlands. The lifting of the receiver to initiate the interaction with the switching centre, and the caller/called-party identification protocols have been omitted from Figure 7.

At the highest level of this goal tree, the intention of the caller is to send the telephone number to the telephone exchange. To achieve (the goal related to) this intention it is converted into a sequence of two sub-intentions, namely to send the area code and thereafter to send the local code. These sub-intentions are further broken down into sub-sub-intentions about sending the individual digits. Finally, the intentions to send digits are translated into observable physical actions, namely the pressing of the number keys on the telephone set.

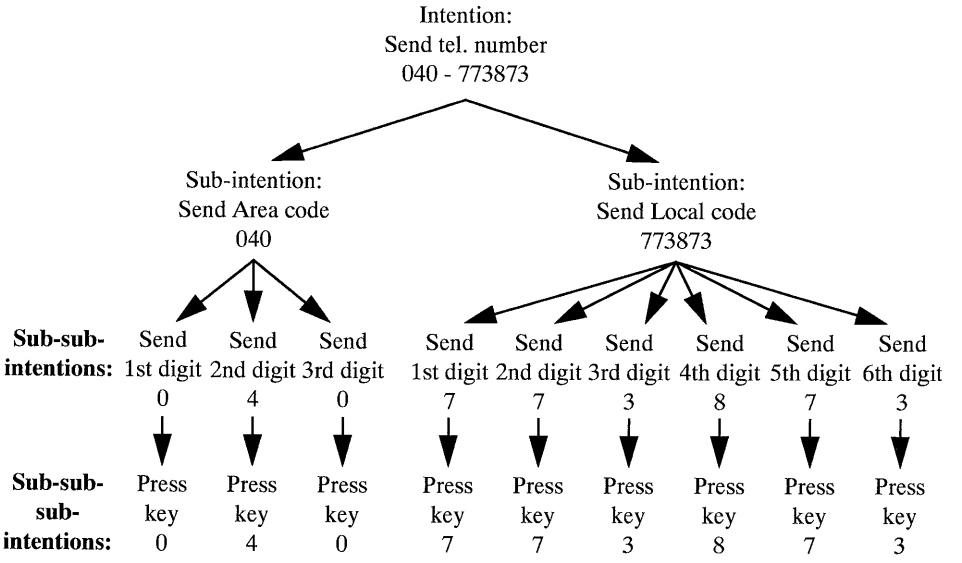


FIGURE 7. Intention-to-action encoding scheme for making a national-long distance call to the IPO. The main intention of the user of the telephone set is indicated at the top. At the lower levels, the main intention is further detailed into sub-intentions to be achieved. The elemental intentions, identified at the lowest level, are converted directly into physical communicative actions.

During communication, the telephone exchange starts interpreting partially received messages and accordingly supplies the caller with early feedback signals. When the handset is lifted from the cradle, the presence or absence of a specific dial tone signals whether a line connection has been made from the telephone set to the telephone exchange. Another tone indicates when the area code has been received and a line to the desired telephone area is made available. After successful reception of an interpretable number and completion of the connection between the telephone sets, the called party's ringing is also conveyed to the caller.

Note that the telephone system performs decoding of the telephone number and building-up of the connection in parallel, so that the caller in fact receives layered feedback about the availability of the lines, whether the called set is free or engaged, etc. If the connection is made fast enough, this more pragmatic feedback, related to the corresponding action, eliminates the need for pure I-feedback about the correct decoding of the received messages.

From the above description it follows that the telephone system provides layered feedback at, amongst others, the level of complete telephone numbers and at the level of area and local code. The beginning and the end of these message units are uniquely marked by dialling tones. Note that feedback relating to the area code is absent in the case of interna-

tional calls. If lines are engaged, this lack of feedback leads to a significant decrease in communication efficiency as the entire telephone number has to be keyed in again. Furthermore, tactile feedback is supplied at the level of keystrokes. The caller is not, however, informed as to what digit has been received or as to the number of digits already received. Input errors made cannot be detected by the caller until the call has been answered, thus creating an inefficient and possibly embarrassing vis-à-vis with the called party. Fortunately, modern touch-tone sets display the selected digits on a small LCD panel.

2.5.3 Layered expectations in telephone-number decoding

In what follows we use the same layered telephone selection procedure for a more detailed analysis of level-specific expectations in message decoding. For decoding the incoming keystroke sequence, the intention-to-action scheme of Figure 7 has to be passed through in the reverse order. At each level of message interpretation, decoding is assumed to supply the semantic type and value of the related message sequences at their input.

More precisely, keystroke values from the keypad {0, 1, ... , 9, #, *} will be the elementary input messages at the keystroke level of the decoding system; see Figure 8. At the first decoding layer, the symbol layer, each incoming keystroke is assumed to be interpreted as either a digital symbol (Sd) type or an alphabetic symbol (Sa) type, briefly indicated by the set {Sd|Sa}. At the next stage, the cluster layer, incoming symbol sequences <Sd|Sa, ... , Sd|Sa> are classified into clusters, namely into an area code cluster (Ca) or local code cluster (Cl), while at the third decoding layer, the telephone-number layer, specific {<Ca,Cl> | Cl} patterns are interpreted as representing a local telephone number (Tl) or a trunk number (Tt).

At each layer, a history of successively interpreted messages is maintained. These histories are in terms of the semantic units recognized at the given layer. So after input of the keystroke sequence <0, 4, 0>, the history at the symbol level reads < Sd, Sd, Sd> and at the cluster level, where an area-code cluster of 3 digits has been recognized, it reads <Ca3>.

Note that by adding the length 3 as a characteristic of the decoded cluster, part of the lower-level information is carried over to the next higher layer. In the ideal case of completely independent layers, this transfer of lower-level knowledge should not be needed. As we do not want to go into this problem here, a higher-level message is described in this example by putting the sequence concerned between brackets and by adding the name of the recognized type in front. In this way each decoding layer attaches specific semantic attributes to the sequence concerned and all lower level information is carried over to the higher layers. So, in our example, Sd(0) indicates that the key stroke 0 has been recognized at the symbol layer as a digital symbol. Similarly, the sequence of 3 digital symbols <Sd(0), Sd(4), Sd(0)> is recognized as Ca3(Sd(0), Sd(4), Sd(0)), an area code cluster of length 3.

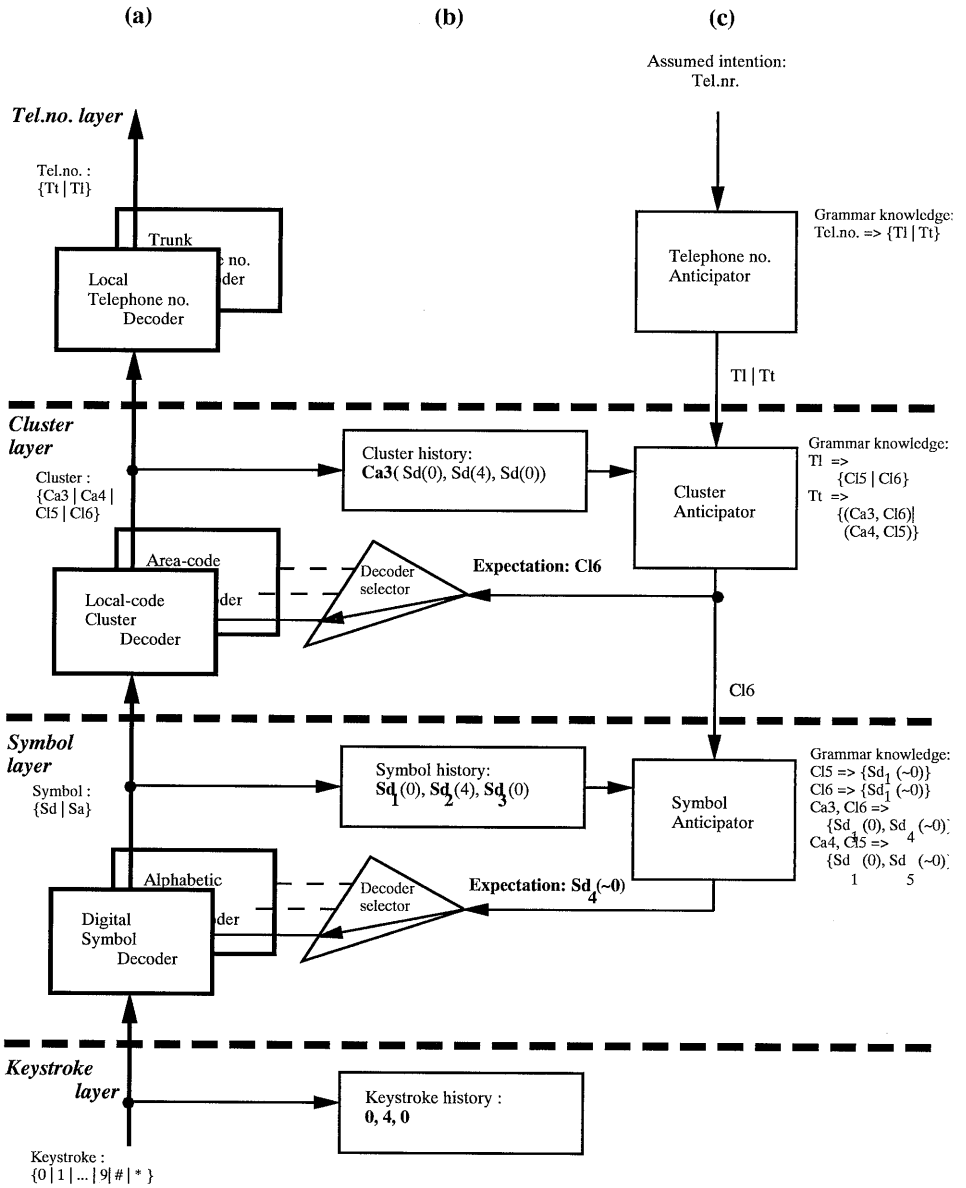


FIGURE 8. Illustration of the relations between (a) layered decoding of an inter-local telephone selection message, (b) layered contextual knowledge about the communication history, and (c) layered message expectations derived from message anticipators. On the left-hand side, the various layers are indicated together with the related message types and permitted values. The right-hand side gives the grammar knowledge available to the anticipators. With help of 1) the assumed message intention, 2) the message grammar, and 3) the current state of the history buffer, the anticipator gives an expectation about the next message element to be received. The following situation is shown: The first three digits have been keyed in and have been interpreted as an area-code cluster of length 3 (Ca3), while the fourth symbol, being a non-zero digit ($Sd_4(\sim 0)$), is expected to be entered. This expectation causes selection of the digital-symbol lexicon. At the cluster level a local code of length 6 (Cl6) is expected, resulting in activation of the local-code decoder. At the telephone-number layer no preference has yet been expressed for the use of the local or trunk-number decoder.

In the model given in Figure 8, each decoding layer is assumed to contain a number of decoders, i.e. a digital and an alphabetic decoder at the symbol layer, a local-code and an area-code decoder at the cluster layer and a local and trunk-number decoder at the telephone-number layer. To clarify the situation, we will assume that each decoder contains a single specific lexicon of entries and related semantic interpretations. If a sequence of message elements at the decoder input matches a lexical entry, the decoder output will give the related semantic interpretations such as grammatical type and semantic value.

At the top of column c in Figure 8, the assumed high-level message intention (communication of a telephone number) is indicated. The telephone-number anticipator contains the (grammar) knowledge that a telephone number can be either a local number Tl or a trunk number Tt (Tel.no. => {Tl | Tt}). If no telephone number has yet been received (an empty telephone-number history), the anticipator will generate its expectation that the next message to be received will be Tl or Tt, i.e. (Tl | Tt). Given from the cluster history, that an area code of length 3 has already been decoded, and given its grammar knowledge of acceptable area/local code combinations (in the, now superseded, Dutch situation the area-code number is 3 or 4 digits long, the total length of area + local code amounts to 9 digits), the cluster anticipator will generate its expectation that a local code cluster of length 6 (Cl6) will follow. Finally, given that 3 digital symbols have already been received and that a local code cluster is to be received next, the symbol anticipator will form its expectation that the fourth keystroke will represent a non-zero digital symbol $Sd_4(\sim 0)$.

These layer-specific expectations are assumed to focus the message-decoding process by activating the decoder concerned via a decoder selector. In the absence of these intention-specific expectations, the various decoders on a given layer are assumed to be activated in fixed (default) order. In our example, the local-code decoder at the cluster level will be activated by expectation of Cl6 and the digital symbol decoder at the symbol level by expectation of Sd(~0). At the level of telephone numbers, no expectations are indicated as yet, so that in the case of incoming signals the lexicons will be consulted in pre-specified order.

By the introduction of context-dependent lexicons/decoders, from which the relevant ones are selected by expectations, the decoder model in Figure 6 is further detailed in Figure 8. Accordingly, each recognizer in Figure 6 corresponds to a decoder selector and related decoders in Figure 8, while the layer-specific message-state signals in Figure 6 are defined in more detail by the introduction of the layered message-history buffers.

2.5.4 Fault recovery in telephone-number selection

At the telephone exchange incoming numbers are verified. An error signal is generated if a non-existing area or local code is received, and not a less harsh, more humanlike, call for clarification. Also, the recovery procedures in the case of non-available lines or non-existing numbers are not very sophisticated. For repair, the caller simply has to send the entire number again. There are no delete last digit or delete last code keys to make recovery more efficient. Note that, in view of the rigid communication context, the occurrence

and resolution of ambiguities is not provided for, at least not for the telephone exchange as a recipient of telephone numbers. For the user, possible ambiguities in, for instance, the tone signals from the exchange have to be resolved by other means, such as consulting a telephone directory.

Fortunately, telephone sets are becoming available nowadays with context-sensitive machine expectations, such as abbreviated dialling and last-number repeat, which improve the efficiency of communication. For enhanced transparency, Figure 3 and 6 indicate that E-feedback should make these machine expectations clear to the user. However, we are not aware up to now of any set in which E-feedback (e.g. about the number of digits still to be entered) is given by the exchange.

2.6 Secondary communication

Now that we have described the transfer of the primary message, discussed layered feedback and layered expectations in further detail and also touched upon fault recovery, it is time to consider more closely the exchange of secondary intentions such as feedback, calls for clarification and related repair messages.

With his 'General Protocol Grammar', in the form of a state-transition diagram applicable on any communication layer, Taylor (1988b) has already described the general structure of the exchange of secondary messages in relation to the primary message. His protocol grammar distinguishes the sending of the primary message by the originator, followed by feedback by the recipient and a final confirmation by the originator. Furthermore, the possibility to skip the sending of feedback and/or confirmation is available . In the case of a detected error and/or ambiguity in the decoding of the primary message or related secondary messages, the protocol grammar provides for error and ambiguity handling. We will now consider these aspects more closely.

2.6.1 Verification and repair

In principle, each message decoder can be provided with a number of input checkers: an alphabetic check can be applied to test whether the incoming message elements are members of the relevant alphabet. A syntax check could verify whether the incoming sequence of message elements possesses the correct grammatical characteristics and, finally, an entry check can test whether the incoming message matches an entry in the lexicon selected. Figure 9 illustrates the elements to be distinguished in a message decoder with verification and repair facilities.

If message verification is not successful and the message cannot be recovered locally by the assumed feedback, error and ambiguity handler (FEA handler) of the given decoder, this handler should initiate a secondary interaction concerning the detected discrepancy between the received message and its expectations by sending a call for clarification. Initiation of feedback is another task of the FEA handler. It has to be decided whether feed-

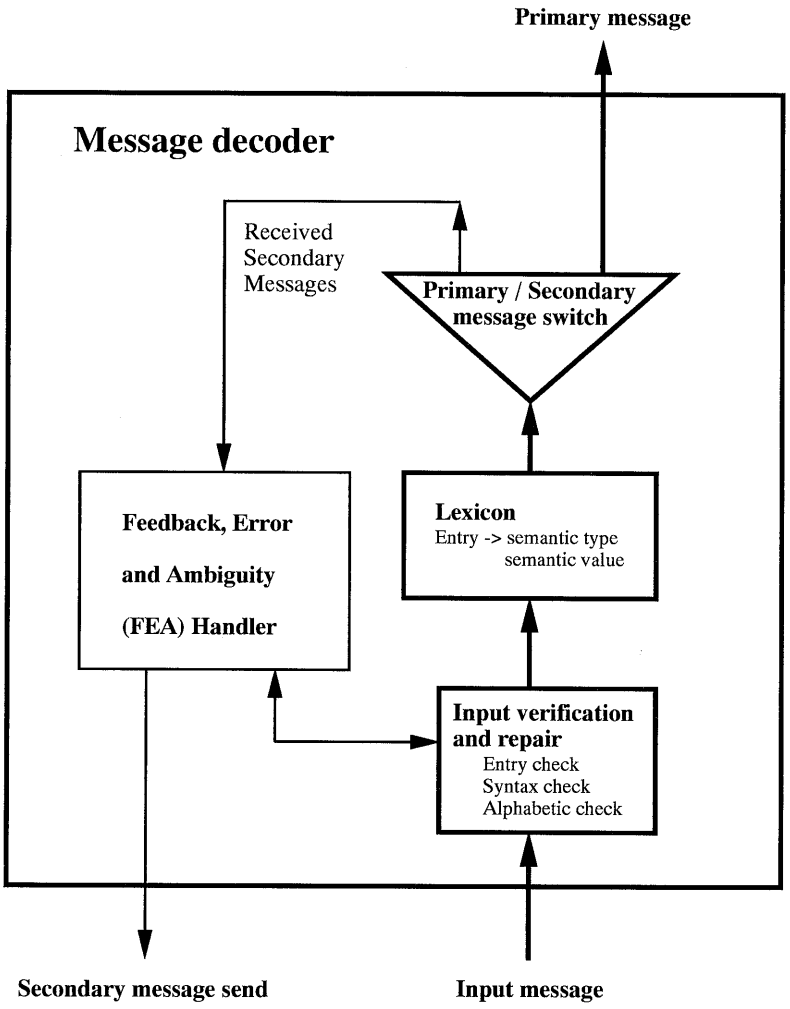


FIGURE 9. Illustration of elements to be distinguished in a message decoder. Besides the lexicon, the message decoder is assumed to contain an input verification and repair process, as well as a system for handling feedback, error and ambiguity messages, namely the FEA handler. After message interpretation by the decoder lexicon, received secondary messages are assumed to be separated from the primary messages by the primary/secondary message switch and sent to the FEA handler. The FEA handler is capable of initiating and ending secondary communication.

back will be given all the time or, for instance, only when recognition has been uncertain because of a minor disagreement with the expected messages. Verification of feedback signals against the original intention of the sender is assumed to be done by the corresponding FEA handler at the sender side.

To distinguish the incoming secondary messages from the primary messages, some sort of a primary/secondary message switch will be needed in the model; see Figure 9. However, the addressing of these secondary messages to the FEA handler at the specific decoding level of the other partner is not trivial.

With regard to our primary/secondary message switch and the related FEA handler, Taylor & Waugh propose a different structure in which the old 'bidirectional protocol loop' has been replaced by two 'unidirectional protocol loops', namely "one for primaries in each direction, with its own feedback system that does not transmit primaries". The result is that partners are assumed to possess at each layer, besides the encoders and decoders for the primary message in both directions, separate coders and decoders for the secondary messages. Accordingly, this proposal also necessitates some switch to change from primary to secondary communication. These differences in structure, however, are not very accessible to external observation. We will not, therefore, go into further detail about them here. What is important, is to recognize that secondary communication constitutes a considerable part of the interaction in human communication and should be applied more extensively in user-system communication to increase ease of use and the efficiency of communication.

Ambiguities arise when multiple interpretations are permitted by the lexicon, or when an incoming message matches an entry present in more than one lexicon of the same layer and expectations do not sufficiently limit these possibilities. In such a situation, the FEA handler might also call for clarification. In general, communication protocols for user-system interaction will be designed in such a way that ambiguities are prevented by design. However, when systems become more complex, it is conceivable that, for instance, temporal ambiguities will arise which have to be solved by secondary interaction.

The dialogue handler of the SPICOS-II continuous-speech information system (de Vet & van Deemter, 1989) can be seen as a first approximation of the layered FEA handlers we have in mind. Figure 10 gives an example of spoken interaction with SPICOS-II, by querying the system's database.

The SPICOS-II dialogue handler permits various sub-dialogues concerning communication problems of different level. For instance, in cases where speech recognition is uncertain, the user's words are echoed and the user is asked to verify whether they have been properly recognized. Syntactic ambiguity at the utterance level is also dealt with by offering the user the competing analyses of the question posed. At the supra-sentence level the dialogue handler is capable of interactively resolving ambiguous anaphoric expressions; see Figure 10. As can be seen in the second part of the interaction in Figure 10, false user presuppositions, incompatible with the information base concerned, can also be handled interactively. This latter type of error has to do with conflict between the domain knowledge of the user and the database system.

a- User: Did Höge meet Ney?
b- SYSTEM: YES, HÖGE MET NEY.
c- User: Did he send a letter to Päseler?
d- SYSTEM: WHO DO YOU MEAN BY 'HE': HÖGE OR NEY?
e- User: O, Höge.

f- User: Did a meeting about human dialogues take place
g- SYSTEM: YES, SIX MEETINGS ABOUT HUMAN DIALOGUES TOOK PLACE!

FIGURE 10. Example of handling supra-sentence ambiguity and resolving false presuppositions on the part of the user. In c) an anaphoric expression is used, i.e. a word that refers to another word. Normally, SPICOS can resolve anaphora by itself, because the system knows what has been said. In this case, however, "he" can refer to either Höge or Ney. The user is therefore asked for clarification in d). In the next question f) the user has wrongly supposed that just one meeting took place. The literal answer of SPICOS could have been "No". However, the dialogue system interprets "a meeting" as "at least one meeting" and dispels the false presupposition by explicit feedback.

2.7 Discussion

2.7.1 Layered protocols

In this paper we emphasized the relation between user friendliness and communication efficiency. Such a trend towards economical behaviour also has been observed by Oberquelle et al. (1983). They mentioned it as one of "the six pragmatic characteristics which apply to both interpersonal and human-computer communication". In line with this, we studied more closely the influence and structure of feedback in user-system interaction. Ideas of Taylor (1988a) concerning "layered protocols for computer-human dialogue" provided a good starting point for this study. His LP-model is based on the observed layering in human perception and performance and recognizes the relevance of feedback at different levels of abstractness for the efficiency of communication.

In agreement with this view, the LP-model fits better to human "information dialogues" (see e.g. Bunt, 1989) than to human chatter. In the same way, politeness is not a necessary ingredient for man-machine dialogues, while "cooperativeness" (see e.g. Kaplan, 1982) certainly is.

The fact that the LP-model has been developed for the analysis of human dialogues as well as for the design and analysis of human-computer interfaces, implies that the machine is assumed to represent some sort of communication partner, with its own intentions. However, as pointed out by Oberquelle et al. (1983), "The computer as a machine

does not have any needs and thus no intentions either. Strictly following programmed procedures it only mirrors its designer's intentions". Because of the limited nature of the knowledge and procedures that can be modelled in a machine in practice, the idea of the machine as an "intelligent assistant" can be approached only to a very small extent. Nevertheless, as we have tried to indicate with our examples, even small steps in that direction can help to make interfaces more efficient and more acceptable for the user.

The LP model differs from earlier ones as given, for example, by Moran (1981), Buxton (1983), Norman (1984) and Nielsen (1986), by the notion that the concepts "what" and "how" play an explicit role at *each* communication level. More precisely, the linguistic concepts of alphabet, syntax, lexicon and semantics should be distinguished on each individual layer of the LP model.

This seems to be in contrast with the earlier layered-interaction models mentioned above, where the naming of the different layers suggests that only a single linguistic concept is involved in each layer. The seven-layer "protocol model" proposed by Nielsen (1986) is characteristic for this more traditional linguistic approach to layered interaction. It distinguishes, besides the lowest "physical layer" concerned with observable physical actions such as the key presses in the interaction with a computer, an "alphabetic layer" related to the characters in the computer command, a "lexical layer" related to command tokens, and a "syntax layer" associated with the command line as the communication unit. In addition to these four "visible layers" three "invisible layers" are distinguished in his model, which "define its (the command's) meaning rather than its form". In other words, the "semantic layer" represents the computer implemented objects and operations, while the "task layer" has to do with larger compounded computer concepts, like macros, scripts, procedures and files. Finally, the top level, named the "goal layer", is related to "real-world concepts", i.e. user goals to be achieved by means of the computer, such as writing a letter. The segmentation applied in this model makes sense in so far as the grain and abstraction of the units grow with the level of communication and their partitioning is familiar to the user. The latter aspect is required to be able to correct these message units individually in the event of communication faults.

Confusion arises, however, when the message units and related layers are associated with specific linguistic concepts such as alphabet, syntax, etc. In Taylor's LP model, the alphabetic, syntactic, lexical and semantic concepts are distinguished at each communication layer: the alphabetic concept represents the set of permitted input elements for the specific decoder stage, the syntactic gives their mutual structure and the lexicon input contains the recognizable combinations of message elements, while their semantics correspond to the lexicon output, viz. the intention assigned by the decoder's recognition process to these higher-level units of message elements. In its turn, the recognized message intention at the decoder's output represents a message-input element for the next higher decoding layer. The reverse ideas hold for the encoding layers.

Furthermore, the number of layers in Taylor's model is not fixed. The differentiation of layers in the LP model is based on the possibility of having secondary communication about the message units of given abstractness.

2.7.2 I-feedback and E-feedback

In his General Protocol Grammar, Taylor (1988b) assumed that the layered communication protocols incorporated layer-specific I-feedback from the recipient, as well as on-line 'help' on request of the originator of the primary message. In this paper we emphasize the relevance of layered E-feedback. This E-feedback can be seen as a specific kind of help, however, initiated by the recipient. It reflects the layered message expectations of the recipient and assists the originator of the primary message in formulating the remaining part of his message.

In fact, message expectations play their role in the intention coding/decoding processes of both partners. In their model of 'intelligent dialogue systems', Edwards & Mason (1989) distinguished, for instance, the concepts of 'Output Message Queue' to contain "pending messages" at the originator's end, and 'Expected Input Queue' to contain "indications of messages expected but not yet received from the dialogue partner". We additionally claim, however, that the message expectations have to be layered. This claim agrees with a recent publication of Taylor & van de Vegte (1992), in which they introduce a "9 element structure of a protocol node", that also makes use of layered message expectations. These message expectations are assumed to influence layered message recognition stochastically. To generate useful context-sensitive receiver expectations, it is essential that messages are structured top-down and start with a global indication of their main intention. Most times a first rough message expectation is already built into the application software by the system designer. Given this first indication of the message to be received in further detail, layered 'semantic' (protocol/grammar) knowledge, as well as 'episodic' (interaction history) knowledge (see Tulving, 1972) in terms of message elements specific to the given layer, is used to generate more detailed, lower-level message expectations.

Of course, semantic knowledge has long been used for command interpretation. The explicit use of episodic knowledge in user interfaces appears to be somewhat less evident. As has been shown, e.g. by Engel, Andriessen & Schmitz (1983) in an experimental Viewdata retrieval system and by Lee (1990) for a number of current systems, the maintenance of an interaction history is an essential ingredient for obtaining effective user support. For the generation of layered message expectations, and also for reference and correction purposes in case of communication faults, we assume the interaction history to be layered as well. For an optimally designed user interface, the possibility of presenting I-feedback concerning the past and current status of interaction and E-feedback concerning the expected future interaction ought to be considered seriously for each layer of communication.

In normal human communication, messages carry just sufficient information relative to the contextual knowledge assumed to be available at the receiver's end. In general, the applied contextual knowledge is very extensive and of various nature (environmental knowledge, partner knowledge, interaction knowledge, topic knowledge, etc.). This remarkable capacity for adaptation to the momentary situation of human (communicative)

behaviour, led Suchman (1987) to talk about "situated action". Note that the context sensitivity applies to both, to the message generation as well as to message interpretation process.

In user-system interaction, however, machine-available context information is fairly limited, with the result that (naive) users have little idea about what is and is not known by the system. In view of the user's uncertainty about the contextual knowledge available to the system, E-feedback increases the system 'transparency' (see, e.g., Maass, 1983). For that reason, both layered E-feedback about the current machine expectations as well as I-feedback about the current interpretation of the incoming message are assumed to be essential ingredients for efficient user-system communication. By identifying the presence or absence of E-feedback and/or I-feedback in a number of existing user interfaces, we have tried to indicate their relevance.

Application selection and mode switches in user-system interaction are simple examples of early information that enables the system to have context-sensitive expectations about what will follow. By means of E-feedback, for instance in the form of successive 'menus', the acceptable lower-level messages are made clear to the user. Also property sheets (also called dialogue boxes) are a clear instance of layered E-feedback: once the machine is aware of the user's higher level goal, e.g. the printing of a file, it indicates by means of a form to be filled in, which lower-level messages it expects to receive for effecting the file printout. Frequently, the data fields in the forms are already provided with default values, thus again increasing communication efficiency, by requiring adjustment only in the case of discrepancies. By prompting the user about the expected messages (commands) to be received, property sheets supply the user with layered help. In the situation in which device prompts correspond exactly to the users current goals, Kieras & Polson (1985) talk of good "task-to-device mapping, meaning that the users task representation in the form of a hierarchical goal structure matches the device command structure. From Suchman's very instructive evaluation of the interaction of a pair of naive users with an advanced copy machine that allowed for top-down control by the user and expectation based command interpretation (Suchman, 1978), it followed that most problems arose from the machine's limited sensitivity for the situation in case of fault handling. Severe mismatches occurred in that case between the user's goal structure and the command structure of the device. Clearly, fault handling is a difficult and less explored subject that certainly requires much more situational background knowledge and related context sensitivity from the side of the machine.

2.7.3 Ease of learning

Ease of learning is another reason for the introduction of layered I- and E- feedback in user-system interfaces. By standardizing the interaction protocols at the lower layers, they will become learned 'skills' for the user. In this way general knowledge about using interfaces becomes segregated from higher-level application-specific knowledge. Note that these acquired lower level communication skills, which can be performed almost auto-

matically (without much conscious effort by the user), still need the related feedback. For instance, in the case of typing, the tactile (and higher level visual) I-feedback remains essential for timely correction of keying errors.

The system's E-feedback helps the user to formulate his message. Once the user has learned to anticipate these message expectations, the E-feedback of the system becomes superfluous at first sight. However, since I- and E-feedback are frequently combined, such as in the pull-down menu's in window-oriented software, the E-feedback cannot be omitted simply. Moreover, removal of E-feedback would disturb the just acquired lower level protocol knowledge.

With the aid of task-action grammars (TAGs), Payne & Green (1986, 1989) described device command structures at the level of simple tasks (routine tasks "needing no external sampling") and made predictions about the ease of learning different (low-level) command structures. In particular, they predicted that consistent structuring of task-action mappings would facilitate learning. When regularities occur in the task-to-action mappings, they can be replaced by a more general rule, thus introducing a higher level of abstraction in the interaction protocol. Accordingly, consistency in the task-to-action mappings is reflected in TAG by a hierarchy of "rule schemas", comparable to the layered encoding structure of the LP model.

TAG's feature grammar enables the specific values of action variables ("feature values") of a given task to be extended to the corresponding task-to-action rule schemas. In this way, it enables the higher-level goals to become known at the lower levels. However, on the basis of their definition of simple tasks, the bottom-up verification of task conditions has been ignored and, being considered as a drawback, closed-loop processing is not supported in their task-action grammar. Above we have indicated the relevance of (bottom-up) feedback for the ease of learning. In defence of their ideas it has to be said, however, that TAG has been developed for the analysis of task languages for consistency and related ease of learning and not so much for the design of interactive message coding/decoding processes.

Hoppe (1988) extended the ideas of TAG by creating a task-oriented parser which analyses the input stream in terms of higher-level task units. In this way he was able to extend TAG to higher levels of message interpretation and build the simple tasks up into more complex tasks. Sequences of user actions could be parsed successfully by the task-oriented parser and Hoppe's program could infer what plan the user was following. In this way (expectation-based) interactive help could be supplied by his program for a small experimental application.

2.7.4 Fault handling

With regard to error and ambiguity correction, an important point is the question of whether or not the handling of a correction procedure is to be considered as a layer- specific affair, to be solved at the layer concerned, as suggested by Taylor (1988a) in his lay-

er-specific General Protocol Grammar. In contrast, it is our opinion that after initiation of a repair action by a layer-specific FEA handler, the handling might be passed on to a higher level of communication. In spoken interaction, for instance, the correction of a received word might be made by uttering a single word, such as the correct name when a person is referred to by the wrong name. But it is more usual to correct such an error by using a proposition, like: "You mean Höge?", a full sentence, or even a new dialogue when major differences in background knowledge need to be corrected. Correction procedures are then initiated by the FEA handler at the level where the error was detected and will be dispatched as a new higher-level communication goal to be solved first.

The latter situation also happens when received E-feedback does not correspond to the message intentions of the originator. In that case the background knowledge of the recipient and/or the originator of the primary message should be adapted first. In relation to Figure 3, where the main signals and functions of a layer-free context-sensitive message decoder were introduced, we already touched upon the relevance of the need for adjustment of the message expectations. Frequently this adjustment will imply that besides these expectations, also the knowledge on which they are based, needs to be corrected. Solutions for this problem of recursive knowledge update are not very well established but they are certainly worth working out further.

In fact, a distinction seems to be needed here between the message communication hierarchy, as realized by the layered message coders and decoders, and a priority structure of intentions to be communicated. To account for related problems of initiative and control in human dialogue, Taylor (1988a) arrived at "a model that allows message passing from level to level not only through the hierarchy of senders and receivers, but also through the connected internal processes, which may not be connected in a hierarchy at all". These internal processes should take care of "thinking" on the part of the human and should take care of "ill-defined aspects of protocols" in the computer. Although little satisfying, it seems that a concept like these internal processes has to be introduced indeed, to account for the above mentioned problem of having to initiate a different level sub-dialogue.

Another aspect of layered I- and E- feedback, to be investigated further, is the question of the conditions under which the supply of feedback might or would be skipped. On the subject of voice I/O, Taylor gives his views about when and how to give voice feedback in aircraft interfaces, e.g. as a function of the reliability of the voice recognizer. The simultaneous communication of multilevel-feedback messages relies on the available physical transmission capacity, as well as on the information-processing capacity of user and system. Delays in feedback because of limited processing capacity may lead to instabilities in communication, as has been pointed out by Taylor (1988a), and should be avoided. The physical transmission capacity of user interfaces can be expanded by combining the visual and auditory channels. How to present the feedback optimally, which levels of feedback should be displayed, how to catch the attention of the user as a function of the relevance of the simultaneously presented secondary messages, and especially how to enable repair of possible errors and ambiguities in communication are open questions, still to be answered in more detail experimentally.

2.8 Conclusions

Guided by the idea that efficiency of communication might be an important determinant of 'user friendliness', we recognized the relevance of early feedback in the layered protocol model of Taylor (1988a). The layered I-feedback in this model enables the originator of a message to verify at an early stage of communication whether his messages are being correctly interpreted by the recipient.

Verification by layered I-feedback is quite functional in human communication as the primary messages exchanged rely largely on all kinds of contextual knowledge which one party assumes the other to have available and which as a consequence are not communicated again. This use of contextual knowledge largely increases communication speed at the risk, however, of interpretation errors due to discrepancies between the assumed and the available knowledge. In analogy with human dialogue, layered I-feedback is also assumed to be important for user-system interaction.

Also, the recipient is usually capable of verifying the incoming message at an early stage. Given some global notion of the message intention, contextual knowledge enables the recipient to predict message elements still to come, by what in human perception is known as 'analysis-by-synthesis' (see, for example, Neisser, 1967) and alternatively in machine recognition is indicated by the 'hypothesize-and-test' strategy (see, for example, Reddy, Erman & Neely, 1973). If the recipient expresses these context-dependent message expectations before the originator has completely uttered his primary message, communication from the originator's end can frequently be limited to simple confirmation. By considering a number of existing user interfaces, we appreciated the relevance of displaying message expectations, here called E-feedback, for the efficiency of the communication of users' intentions, as well as for the transparency of the system to its user. Accordingly, we extended Taylors LP model with the possibility of layered E-feedback that comes from the recipient of the primary message. Given a more global, higher-level assumption about the intention of the primary message to be received, and given the current state of interaction, layered 'anticipators' forecast the possible message elements to be received next. For that purpose each decoding layer in the LP model has been attributed a message anticipator and a temporal store containing the current state of the communication history in terms of the corresponding message units.

To summarize, we claim that it follows from our analysis of existing user interfaces that extension of the LP model with layered E-feedback holds. For efficient, easy-to-use systems, the user-interface designer ought to consider, for each possible level of interaction, the possibility of presenting I-feedback as well as E-feedback to the user. Moreover, if this feedback is given, layer-specific correction procedures should be made available as well.

We gratefully acknowledge the I- and E- feedback received from Martin Taylor and two anonymous referees on the first version of this paper. Their helpful comments have improved the manuscript considerably.

2.9 References

Barwise, J. & Perry, J. (1983). *Situations and Attitudes*, pp. 32-45. Cambridge, MA: MIT Press.

Beun, R.J. (1989). *The Recognition of Declarative Questions in Information Dialogues*. Tilburg: Doctoral Thesis, Tilburg University, The Netherlands.

Boring, E.G. (1950). *A History of Experimental Psychology*, pp. 147-149. New York: Apple-Century-Crofts.

Bösser, T. (1987). *Learning in Man-Computer Interaction*. Berlin: Springer Verlag.

Bunt, H.C. (1988). On-line interpretation in speech understanding and dialogue systems. In H. Niemann, M. Lang & G. Sagerer, Eds. *Proceedings of the NATO ASI: Recent Advances in Speech Understanding and Dialog Systems*, pp. 349-395. Berlin: Springer Verlag.

Bunt, H.C. (1989). Towards a dynamic interpretation theory of utterances in dialogue. In H. Bouma & B.A.G. Elsendoorn, Eds. *Working Models of Human Perception*, pp. 419-455. London: Academic Press.

Bunt, H.C., Leopold, F.F., Muller, H.F. & van Katwijk, A.F.V. (1978). In search of pragmatic principles in man-machine dialogues. *IPO Annual Progress Report*, **13**, pp. 94-98.

Buxton, W. (1983). Lexical and pragmatic considerations of input structures. *ACM SIGGRAPH Computer Graphics*,**17**, pp. 31-37.

van Deemter, C.J. (1991). *On the Composition of Meaning*. Amsterdam: Doctoral Thesis, University of Amsterdam, The Netherlands.

Edwards, J.L. & Mason, J.A. (1989). The structure of intelligence in dialogue. In M.M. Taylor, F. Nel & D.G. Bouwhuis, Eds. *The Structure of Multimodal Dialogue*, pp. 85-105. Amsterdam: North-Holland.

Engel, F.L., Andriessen, J.J. & Schmitz, H.J.R. (1983). What, where and whence: Means for improving electronic data access. *International Journal of Man-Machine Studies*, **18**, pp. 145-160.

Hoppe, H.U. (1988). Task-oriented parsing - A diagnostic method to be used by adaptive systems. *Proceedings CHI'88 Human Factors in Computing Systems*, pp. 241-247. New York: ACM.

Kaplan, S.J. (1982). Cooperative responses from a portable natural language query system. *Artificial Intelligence*, **19**, pp. 165-187.

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

Lee, A. (1990). A Taxonomy of Uses of Interaction History. In S. MacKay, Ed. *Proceedings of Graphics Interface '90*, pp. 113-122. Toronto, Ontario: Canadian Information Processing Society.

Levelt, J.M. (1983). Monitoring self-repair in speech, *Cognition*, 14, pp. 41-104.

Maass, S. (1983). Why systems transparency? In T.R. Green, S.J. Payne, & G.C. van der Veer, Eds. *The Psychology of Computer Use*, pp. 19-28. London: Academic Press.

Marslen-Wilson, W.D. (1980). Speech understanding as a psychological process. In J.C. Simon, Ed. *Spoken Language Generation and Understanding*, pp. 39-67. Dordrecht: Reidel.

Miller, G.A., Heise, G.A. & Lichten, W. (1951). The intelligibility of speech as a function of the context of the test materials. *Journal of Experimental Psychology*, **41**, pp. 329-335.

Miller, G.A. & Isard, S. (1963). Some perceptual consequences of linguistic rules. *Journal of Verbal Learning and Verbal Behavior*, **2**, pp. 217-228.

Moran, T.P. (1981). The command language grammar: A representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies*, **15**, pp. 3-50.

Neisser, U. (1967). *Cognitive Psychology*, pp. 193-198. New York: Appleton-Century-Crofts.

van Nes, F.L. & van Itegem, J.P.M. (1990). Hidden functionality in the usage of car-radio controls. *IPO Annual Progress Report*, **25**, pp. 101-112.

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

Norman, D.A. (1969). *Memory and Attention: An Introduction to Human Information Processing*, pp. 49-50. New York: Wiley.

Norman, D.A. (1981). Categorization of action slips. *Psychological Review*, **88**, pp. 1-15.

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

Oberquelle, H., Kupka, I. & Maass, S. (1983). A view of human-machine communication and co-operation. *International Journal of Man-Machine Studies*, **19**, pp. 309-333.

Payne, S.J. & Green, T.R.G. (1986). Task-action grammars: a model of the mental representation of task languages. *Human-Computer Interaction*, **2**, pp. 93-133.

Payne, S.J. & Green, T.R.G. (1989). The structure of command languages: An experiment on task-action grammar. *International Journal of Man-Machine Studies*, **30**, pp. 213-234.

Reddy, D.R., Erman, L.D. & Neely, R.B. (1973). A model and a system for machine recognition of speech. *IEEE Transactions on Audio and Electroacoustics*, **AU-21**, pp. 229-238.

Schegloff, E.A. (1980). Preliminaries to preliminaries: Can I ask you a question. *Sociological Inquiry*, **50**, pp. 104-153.

Suchman, L.A. (1987). *Plans and Situated Action: the problem of human-machine communication*. Cambridge: Cambridge University Press.

Tanenbaum, A.S. (1981). *Computer Networks*, pp. 10-21. Englewood Cliffs, N.J.: Prentice-Hall International Inc.

Taylor, M.M. (1987). Layered protocols in voice interaction with computers. In *Information Management and Decision Making in Advanced Airborne Weapon Systems*. NATO Advisory Group for Aerospace Research & Development, Aeromedical panel CP-414.

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

Taylor, M.M. (1988b). Layered protocols for computer-human dialogue. II: Some practical issues, *International Journal of Man-Machine Studies*, **28**, pp. 219-257.

Taylor, I. & Taylor, M.M. (1983). *The Psychology of Reading*, pp 165-207. New York: Academic Press.

Taylor, M.M. & van de Vegte, J. (1992). Strategies for speech recognition and understanding using layered protocols. In P. Laface & R. de Mori, Eds. *Speech Recognition and Understanding. Recent Advances*, pp. 395-400. Berlin: Springer-Verlag.

Taylor, M.M. & Waugh, D.A. (1993) Dialogue analysis using layered protocols. *Proceedings of the Sorrento Workshop on Pragmatics*. Sorrento, Italy, 3-7 June 1991. (In press.)

Taylor, M.M. & Waugh, D.A. (1993) Principles for integrating voice I/O in a complex interface. *AGARD Avionics Panel Symposium: Advanced Aircraft Interfaces: The Machine Side of the Man-Machine Interface*, Madrid, Spain, May 16-22, 1992. (In press.)

Tulving, E. (1972). Episodic and semantic memory. In E. Tulving & W. Donaldson, Eds. *Organization of Memory*, pp. 381-403. New York: Academic Press.

de Vet, J.H.M. & van Deemter, C.J. (1989). The SPICOS-II dialogue handler. *IPO Annual Progess Report*, **24**, pp. 105-112.

Chapter 3

Layered Protocols: hands-on experience[1]

Abstract

An assessment is presented of the benefits and limitations of the Layered Protocols model for the analysis and design of user interfaces in the field of consumer electronics. In the assessment a user interface of an existing digital audio recorder which is only partly in line with the LP model is compared with an interface designed according to the model. The observed differences in usability between the two interfaces are mainly caused by deviations from the LP model. It turned out that especially the learnability of an interface is positively influenced by a layered organization of user-system interaction in combination with high-quality E- and I-feedback and optimum similarity between interaction protocols.

3.1 Introduction

User-system interaction models and user interface design guidelines can be useful tools for designing better conceived and more usable interfaces (Marshall, Nelson & Gardiner, 1987; HUSAT, 1988). They can be of help in structuring the interaction between the user and the system and in explaining the role of the information exchanged. Ways of increasing usability are indicated on the basis of implicitly or explicitly incorporated cognitive principles. In this way, validated interaction models and guidelines can help in reaching a basic level of usability while diminishing the need for extensive user testing.

In this paper the validity of a specific user-system interaction model is assessed: the Layered Protocols (LP) model (Taylor, 1988a). The assessment is carried out in the field of consumer electronics: digital audio recorders. The application area of consumer electronics differs from the professional setting with respect to training. Users operating a digital audio recorder have usually had no formal instruction or coaching. Therefore, more weight is attached to learnability than to efficiency as a usability criterion (Nielsen, 1993).

The LP model is based upon the cognitive principle that humans use superimposed layers of abstraction in perception and performance. From this principle the LP model arrives at an architecture for structuring user-system interaction. The model explicitly addresses the

---

1. This chapter has been published as a paper by Eggen, J.H., Haakma, R. & Westerink, J.H.D.M. (1996) in the International Journal of Human-Computer Studies, volume 44, pp. 45-72.

user's and the system's contribution to the interaction and the way in which they relate to each other. In this paper guidelines indicating potential usability consequences are derived from the model. That way the model can be validated.

The LP model was chosen because it seemed somewhat more applicable in consumer electronics than some alternative models. Other models like Cognitive Complexity Theory (Kieras and Polson, 1985) and Norman's stages and levels (Norman, 1984) also take a cognitive approach. However, the LP model is more explicit in the implications of the principles for structuring user-system interaction. The GOMS and the keystroke model (Card, Moran & Newell, 1983) are not appropriate because they emphasize efficiency of interaction. Learning is also addressed by Task Action Grammars (Payne & Green, 1986). However, its focus is on structuring the user's part of the interaction. The issue of feedback is disregarded. For a more elaborate discussion of this issue see Engel & Haakma (1993).

The LP model will be validated in the following way. First, the principal concepts of the LP model will be described and some guidelines indicating usability consequences will be derived from the model. After that, these concepts will be made operational in the analysis of an existing interface in terms of the LP model. On the basis of the results of this analysis some interface properties will be indicated as causes of potential usability problems. Next, it will be established whether the LP model is specific enough to guide user interface design by creating a new user interface for a functionally equivalent recorder. The design will be based on the LP model. Subsequently, the two interfaces will be evaluated with users. The results of the experiments will reveal the interface with the highest usability. Finally it will be discussed to what extent the usability differences are accounted for by deviations from the LP model and to what extent the LP model is found to be incomplete in covering user interface aspects influencing usability.

3.2 The Layered Protocols model

3.2.1 An overview of the model

The Layered Protocols model is based on the idea that interaction between communicating partners takes place in a series of layers that represent different levels of abstraction. Taylor (1988a) gives persuasive evidence for this claim. As a consequence the LP model organizes user-system interaction as a hierarchically structured framework of communication loops, each operating according to its own interaction protocols.

Figure 1 shows the elementary communication loop that forms the basis of the LP model. The interaction is started by a user with a particular intention in mind. In order to let the system know this intention, the user has to encode it into a message, for example by pressing a sequence of buttons. Upon receiving an action sequence the system decodes this message and arrives at an interpretation. This interpretation will cause the system to take actions aimed at fulfilling the user's intention. The system also provides feedback by en-

coding either its interpretation of the message received or the results of the actions taken. The communication loop is closed when the user interprets the system's responses and checks to what extent the original intention has been satisfied.

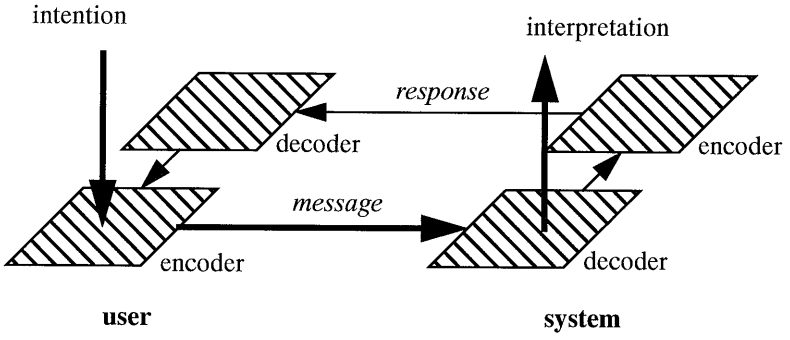


FIGURE 1. Basic communication loop of user-system interaction.

As a consequence of the observation that humans use layers of abstraction in perception and performance, the Layered Protocols model organizes user-system interaction in a hierarchical way (see Figure 2). On the user's side the incoming intention is encoded into sub-intentions at each level. Each sub-intention constitutes an incoming intention for the layer below. In this way the user's overall intention is successively encoded into sub-intentions until it has been transformed into a sequence of physical actions. The system decodes the message it receives in reverse order. At each level it will try to make an interpretation to the next level up on the basis of sub-intentions recognized in the layer below. Feedback is supplied on all (intermediate) interpretations at every level. Those feedback messages may be directly transmitted to the user (e.g. they may be fully displayed on a computer screen). Or they may be communicated in bits and pieces requiring the use of another (layered) protocol directed the other way, that is from system to user.

In order to facilitate effective interaction between the user and the system, the corresponding coding and decoding processes have to employ the same interaction protocol. Taylor (1988b) uses a state-transition diagram to describe a general protocol grammar. The core of this grammar, linking user messages to system feedback optionally followed by an acknowledgment, is shown in Figure 3.

This core grammar suffices when mutual agreement on the protocols is guaranteed. However, in order to set up robust communication in practice, interaction protocols additionally have to indicate how to deal with communication failures. Therefore Taylor's general protocol grammar indicates several expansions of this basic protocol structure to enhance robustness of communication. The two most important extensions facilitate:

- adjustments: when the system's interpretation deviates from the user's intention in details, the user is offered the opportunity to adjust the interpretation;

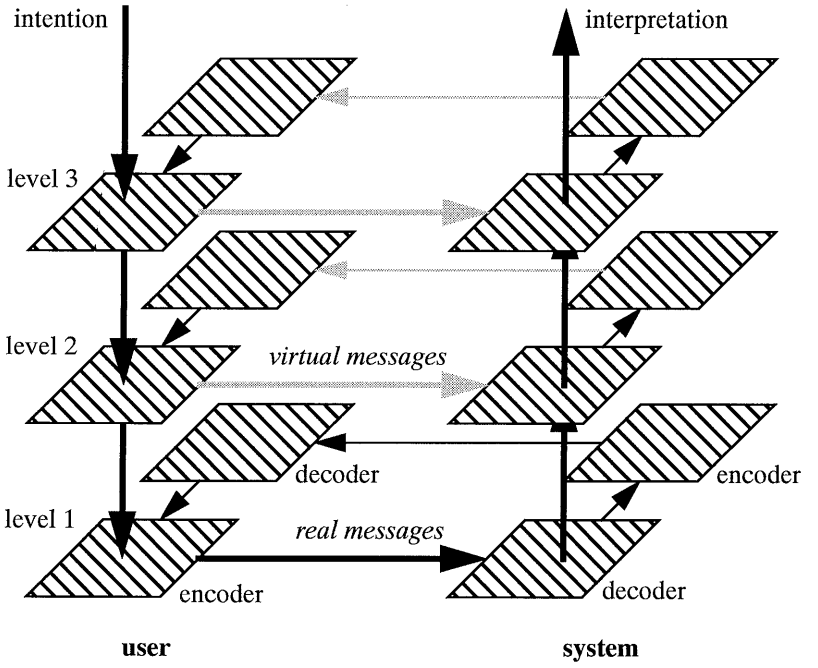


FIGURE 2. The Layered Protocols model of user-system interaction.
The transfer of an intention from user to system is indicated by thick arrows. Via successive stages the main intention is encoded into a message at the lowest level. The message is transmitted to the system via a physical communication channel. The system in turn successively decodes this message into the user's higher-level intention. Feedback at each level (thin arrows) enables the user to verify the system's interpretations at all stages. The grey arrows indicate the exchange of virtual messages between the user and the system in successive layers. A virtual message is considered to pass on the contents of a real message communicated by the underlying layers.

- error and abort handling: for dealing with situations in which the interpretation has to be dropped, i.e. when the system's interpretation does not match the user's intention by far, when the system cannot carry out the user's intention or when the user has second thoughts.

In the original LP model the sole purpose of feedback is to provide users with information enabling them to check to what extent their intentions have been carried out. In Engel & Haakma (1993) it was recognized that users not only require information on the system's *interpretation* of their messages (I-feedback). They also need information on how to formulate their messages, i.e. on what information the system *expects* to receive from the user (E-feedback).

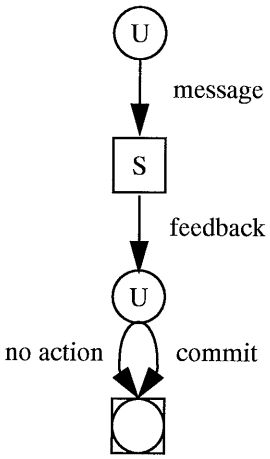


FIGURE 3. The core protocol structure.
The User (U) sends a message to the system, the System [S] interprets the message and gives feedback on its interpretation. After that the User can take this interpretation for granted and proceed with the next message or an explicit commitment may be required from the user.

E-feedback assists users in coding their messages. E-feedback has the same layered structure as I-feedback because E-feedback is required for guiding the users in each coding step. E-feedback differs from I-feedback with respect to the moment it is needed: E-feedback is helpful before users code their intentions whereas I-feedback is needed after the message has been communicated. A protocol structure incorporating E-feedback is presented in Figure 4. The difference in timing between E- and I-feedback is clearly indicated.

3.2.2 Guidelines

The mere fact that an interaction model is based on some cognitive principle does not guarantee that it can be successfully used to guide user interface analysis and design. Confidence in the usefulness of the model in practical situations is increased when sensible user interface guidelines can be deduced from the model. Guidelines also make the model more operational. Only when usability consequences are indicated does it become possible to validate the predictions of the interaction model. Therefore this section lists some testable guidelines for structuring user-system interaction that can be derived from the LP model.

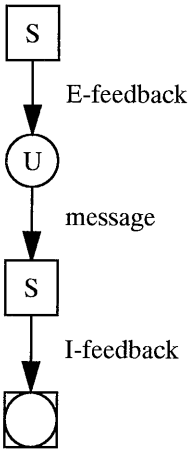


FIGURE 4. A protocol structure incorporating E-feedback.
The System [S] provides the user with E-feedback on the options available, the User (U) composes a message on the basis of this information, the System interprets the message and gives I-feedback on its interpretation.

- *E-feedback*
  E-feedback has to be provided at the various levels of the interaction. In this way the expectations of the system with respect to the message to be received from the user are reflected and users are thus assisted in coding their intentions. E-feedback should give rise to expectations on the part of the users on the system's interpretation of messages.
  For example, at the lowest layer the presence of a button may invite users to press it, while the label of the button may give rise to higher-level expectations on the effect that pressing that button may have.
  When E-feedback is lacking it is assumed that novice users initially have problems in predicting the effect of a message and in composing the appropriate one.

- *I-feedback*
  The LP model stresses the importance of I-feedback. Every user action can in principle have an immediately observable effect. On account of the layered structure of the LP model the system's interpretation is immediately signalled back at the highest level possible: an interpretation is made as soon as the relevant information is available. The provision of immediate I-feedback at the successive level enables users to detect unintended interpretations at an early stage. Without the layered feedback there would be a good chance of errors remaining unnoticed until later on, when it would be more difficult to correct them. Immediate I-feedback also facilitates learning because users can immediately relate their messages to the corresponding system responses.
  Suppose that to a computer with a command-line interface 'del <file name> <CR>'

has the meaning of delivering the file via e-mail to some default address. When users with the intention of deleting some file type 'del <file name> <CR>', they would only notice the unintended effect after the command had been executed from the system response 'Delivered <file name> to <mail address>'. If the system were to provide intermediate feedback, for example by showing a mailbox on the screen after 'del_' has been typed and a trash can after 'rm_', users would be able to detect the misunderstanding and initiate error repair at an early stage.
When I-feedback is lacking users are assumed to have problems in establishing the effect of an action. This may make it hard to learn how to operate the interface and to determine to what extent a particular intention has been satisfied.

- *Error and abort handling*
  Error and abort handling facilities are required to prevent that situations arise in which users are trapped in an unintended situation and are forced to proceed in an undesired direction. The availability of these facilities at different levels enables users to correct parts of an interpretation selectively. Error and abort handling facilities can be offered explicitly to users, e.g. recording music onto an audio cassette can be interrupted by pressing the stop button. They can also be incorporated more implicitly by offering functions that undo each other's effects. E.g. in the case of a CD player, the effect of pressing the next button can often be neutralized by pressing the previous button.
  When error and abort handling are not available at the various levels, it is more difficult to correct interpretations. Exploration of the system is also more difficult because users have to proceed in situations in which they want to backtrack and explore alternatives.

- *Similarity of protocols*
  Each layer has its own interaction protocol that interprets the messages coming from the layer beneath. Therefore, the protocols at the different levels are only loosely coupled so as to enable independent protocol analysis: lower-level protocols can be designed and analyzed more or less independently of higher-level protocols. Also, a lower-level protocol can support underlying interaction for several higher-level protocols. For example, in the case of audio cassette recorders the same interaction protocol is used for indicating the tape position at which playing should start and for indicating where a new recording should start.
  When interaction is structured by employing the same underlying protocols for different tasks the internal consistency is increased, which makes it easier to learn how to operate the system.

In short, it is assumed that user interfaces designed according to the LP model will excel in early detection and repair of communication failures. Learning is facilitated because users can immediately link their actions to observable system responses and because of internal consistency. Guidance in the form of E-feedback assists users in coding their messages.

3.3 Analysis of an existing user interface

The LP assessment will start with the analysis of an existing digital audio recorder in order to see how well its interface can be described in terms of the LP model. The analysis will result in the identification of potential usability problems.

The analysis was performed using a commercial digital audio tape (DAT) recorder. The DAT recorder offers functionality that is characteristic for digital audio recorders in general. The frequently used features comprise playback and tape transport. The recorder also allows more complex operations, including programming and skipping tracks, recording and tape editing (e.g. merging two tracks).

3.3.1 Recording using the DAT recorder

In order to give an impression of what the interaction is like, a scenario will be elaborated in which a user has the intention of recording a new track from a CD player immediately after track number 3 and thus overwriting track 4. All the relevant buttons on the front panel and the indicators on the display of the DAT recorder are shown in Figure 5.

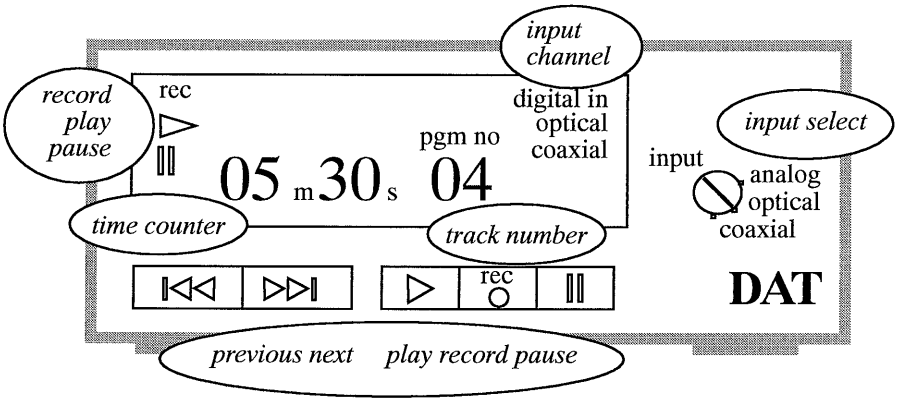


FIGURE 5. Impression of the front panel of the DAT recorder.
Only those buttons, knobs and feedback indicators are presented that are relevant to the recording task described in the text. The record task requires the use of five buttons and a knob. The knob is used to select the audio input channel. The selected input channel is also indicated on the display. A time counter indicates the position of the head on the tape. The current track number is also shown. Three other indicators show whether the recorder is recording, playing or pausing. The words in italics are explanations not appearing on the DAT recorder itself.

After the tape has been inserted into the DAT recorder the user has to check on the basis of the indicators in the display whether the intended source has been selected, whether the tape head is at the correct position and whether the recorder is in record-pause mode. If this is not the case, these settings have to be adjusted.

If the tape head is not at the beginning of track 4, the user first has to transport the tape by repeatedly pressing the previous or next button. After one of these buttons has been clicked, the recorder starts making winding sounds, showing a changing counter and most often a new track number.
Once the tape head has reached the correct position, the user has to press the record button and will then see the record and pause indicators light up. The DAT recorder is now in record-pause mode.
The user also has to select the correct input channel for the music using the input select button. The selected channel is also shown on the display. The correct input channel can be selected while the other adjustments are being made.
When all the adjustments have been made, the user can start the recording by pressing the play or the pause button. The recorder shows it has started recording by switching the pause indicator off and the play indicator on, while the tape counter starts increasing.

3.3.2 LP description of the recording function

The interaction as described in the scenario can be structured in three layers, each characterized by its own protocol. Figures 6, 7 and 8 show the protocols involved.

The top-level interaction protocol is related to the overall goal of recording. It is a single-step protocol in which users have to pass the message 'to record from a particular input channel starting at a particular tape position'. This top-layer message is an interpretation of the bottom layer message 'the user has pressed the play or pause button' in combination with some settings that can be adjusted using the middle layer's protocol. That the recording is in progress can be concluded from the increasing counter and the lighting up of the record and play indicators while the music being recorded can be heard. This information therefore functions as I-feedback. The settings that influence the precise effect of pressing the play or pause button are the input channel indicator and the tape head position indicated by the track number and the time counter. The recorder has to be in record-pause mode, indicated by the record and pause indicator, when the recording is started. All the information on the relevant settings in combination with the play and pause symbols are classified as E-feedback since it should make users realize that recording will start after the play or the pause button has been pressed.

The middle layer is concerned with adjusting the settings relevant to the top layer. This layer consists of two parallel protocols: one for changing the selected input channel and another for changing the tape head position and entering the record-pause mode.
The middle layer's message to change the input channel is an interpretation of the bottom layer's message that the user has turned the rotary input select knob. The selected input channel is shown on the display by the input channel indicator. This information corresponds to the position of the knob. The input channel indicator in combination with the knob position therefore serves as I-feedback. The labels around the knob indicate what input channel will be selected after the knob has been turned. This information therefore serves as E-feedback.
The DAT recorder requires that the tape head position be adjusted before the record-pause

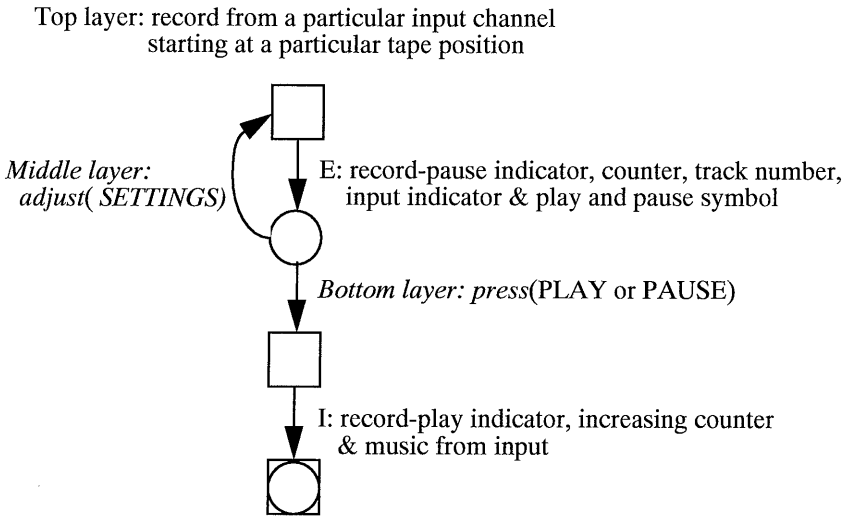


FIGURE 6. Top-layer interaction protocol showing how the message 'to start recording from a particular source on a particular track' is constructed from lower-level messages, showing the E-feedback to get users to do it that way and I-feedback to reflect the system's response.

mode is entered. The middle layer message to adjust the tape head position is an interpretation of the bottom layer message that the user has pressed the next or previous button. The track number in combination with the counter indicates the current tape head position and therefore serves as I-feedback. E-feedback is provided by the previous and next symbols on the buttons. The middle layer message to enter record-pause mode is an interpretation of the bottom layer message that the user has pressed the record button. The record and pause indicators serve as I-feedback while the symbols on the record buttons serve as E-feedback.

The third and bottom layer involves turning the knobs and pressing the buttons. It generates the bottom message that the user has pressed some button or turned some knob the moment it detects that the user executes such a physical action. The presence of the knobs and buttons should make users realize that they can turn or press them and they therefore function as E-feedback. I-feedback is provided via the tactile and kinesthetic senses, in combination with a clicking sound of the button's mechanism. It should make users aware that the button has been pressed or the knob been turned.

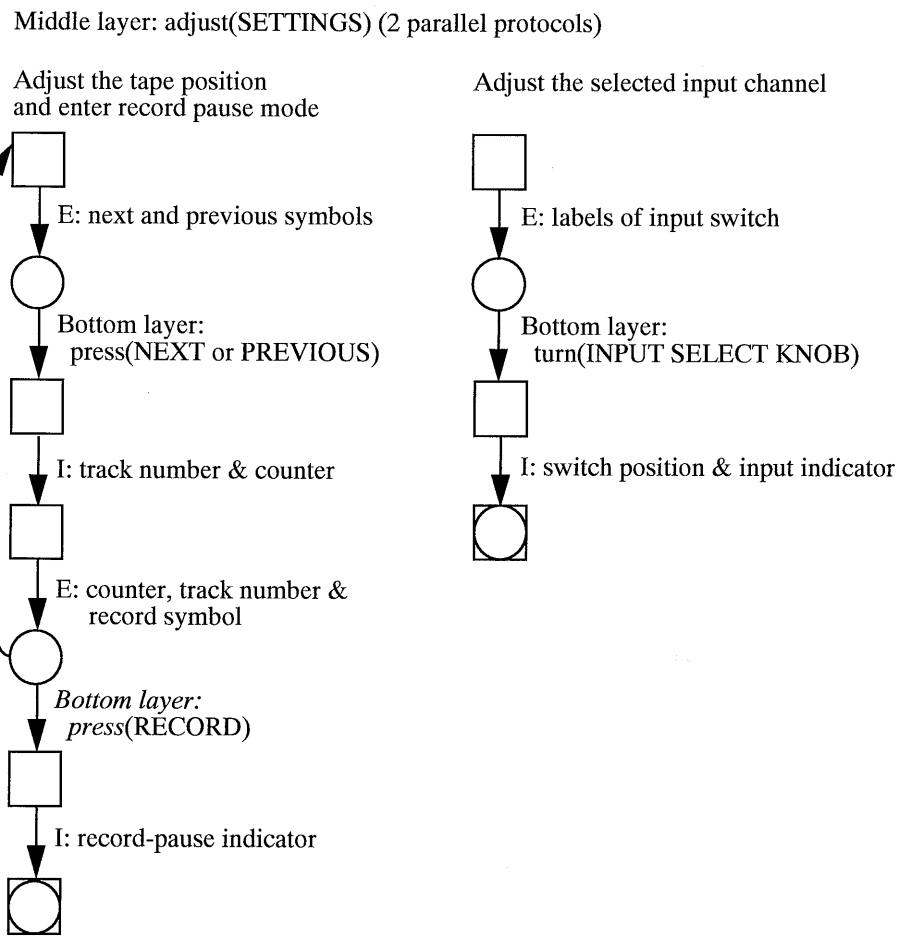


FIGURE 7. Middle-layer protocol for adjusting the recorder's settings.

Apart from on the recording functionality, the LP analysis also focused on other functions, such as programming and tape editing. On the whole, it turned out to be complicated and laborious to describe the interaction between the user and the DAT recorder.

The fact that the DAT recorder gives no I-feedback of a more abstract nature at the top level (e.g. no information is provided on the tape contents) made the analysis *complicated.* It would have been impossible to distinguish between the top layer and the middle layer by solely looking at the feedback supplied. In this analysis the way in which the user's part of the interaction is organized is used as a guide in distinguishing between layers. It then becomes clear that the top layer codes its feedback solely in terms used at the level beneath.

The work was also *laborious* because of the great number of different protocols used in

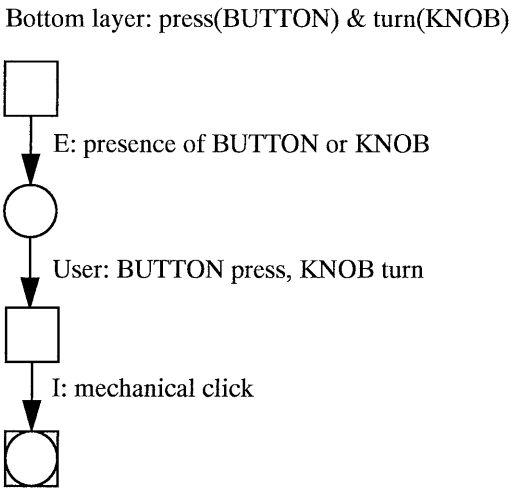


FIGURE 8. Bottom-layer protocol for pressing buttons and turning knobs.

the DAT interface. Almost each function has its own protocols, which differ from the protocols applied for other functions. The only major exceptions are the transport controls for tape-head positioning, which are used in both the recording and the editing functionality.

3.3.3 Usability considerations

It will now be discussed to what extent the DAT recorder was found to follow the guidelines derived in section 3.2.2. On the basis of these guidelines some potential usability problems will be indicated.

- *E-feedback*
  E-feedback is offered in the form of button labels on the front panel. The system settings relevant for predicting the interpretation of a user message are shown on the display.
- *I-feedback*
  The immediate I-feedback provided by the DAT recorder often consists of changes in the status information on the display. At the highest level no I-feedback of a more abstract nature is given about the effect of the current settings on the recording being made. Instead, the users have to infer these effects from the I-feedback given at the middle level. Therefore users may have difficulty in establishing whether the tape has been affected and if so, how. This can have the consequence that users are slow in learning how to use the recorder and in detecting communication failures.

- *Error and abort handling*
  The protocols generally contain various possibilities of repairing errors and aborting the interaction. Functions that undo each other's effects are provided, e.g. 'next' and 'previous'. Some explicit abort facilities are also provided such as a 'stop' button.
- *Similarity of protocols*
  There is not much similarity between the interaction protocols used for selecting different functions. The interaction is structured according to a great number of different protocols. This again may slow down users in learning how to control the recorder.

In short, it is assumed that learning how to operate the DAT recorder is slowed down by the lack of I-feedback at the top level and by insufficient similarity between the applied interaction protocols.

3.4 Design of a new user interface

A new user interface offering functionality equivalent to that of the DAT recorder was designed. The interface was a prototype for a Digital Compact Cassette (DCC) recorder. Like the DAT recorder, a DCC recorder is a type of digital audio recorder.

The DCC prototype was explicitly designed to be in line with the LP model. The interaction is structured in three layers. At the top level functions operate on the tape contents. To visualize this, the tape is graphically displayed in the lower part of the screen. By filling in a form, presented in the upper part of the screen, users can select the function that matches their intentions (see Figure 9). The middle layer provides the means for changing the contents of the form. At the bottom level the physical interaction, consisting in pressing buttons, takes place.
Three forms were designed, each addressing a different type of user intention: one form for *programming*, one for *recording* and one for *editing* functions. These different functions are selected by using the same underlying form-filling protocol.

3.4.1 Recording using the DCC prototype

In order to give an impression of the interaction, a scenario similar to that of the DAT recorder will be elaborated (see section 3.3.1): a user has the intention of recording a new track from a CD player immediately after track number 3 and thus overwriting track 4. All the relevant buttons on the front panel of the DCC prototype are shown in Figure 10.

First, the user has to select the record form by pressing the 'record' button on the front panel of the digital recorder. The record form containing four fields appears on a display (see Figure 9). Next, the user may wish to change the contents of these fields in such a way that they correspond to the recording intention. With the aid of four cursor keys labelled 'move' a thick black frame can be moved over the record form to indicate which field must be adjusted, e.g. the 'from' field. The contents of the field indicated by the frame can be changed by turning the knob labelled 'change'. The user can feel every turn

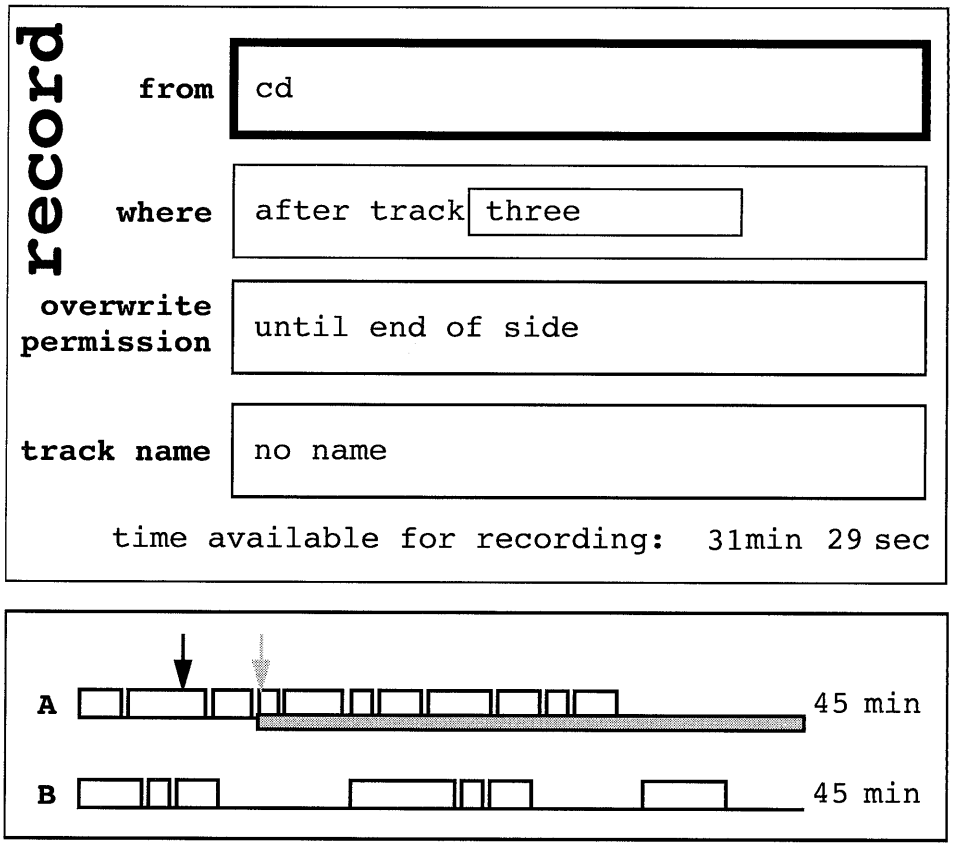


FIGURE 9. The screen of the DCC prototype.
The upper part of the screen shows the form used for recording. One of the fields in the form is surrounded by a thick black frame. A graphical representation of the tape contents is presented in the lower part of the screen. The tracks on the two sides of the tape are presented as white boxes on two lines. The black arrow indicates the tape head position. Additional information on the interpretation of the form is supplied by the dark grey bar indicating what part of the tape is reserved for the recording, the grey arrow indicating where the recording will start and the field indicating the time available for recording.

of the rotary knob and can see that the next or previous option is displayed in the selected field, e.g. from 'radio' to 'cd'. In this way the user can scan all the options available for the field. The recorder gives indications of the interpretation of the form: a grey bar indicates what part of the tape is reserved for the recording and the time available for recording is shown to allow the user to check whether the reserved amount of space will be sufficient. After the record form has been adjusted, the user has to press the 'go' button to

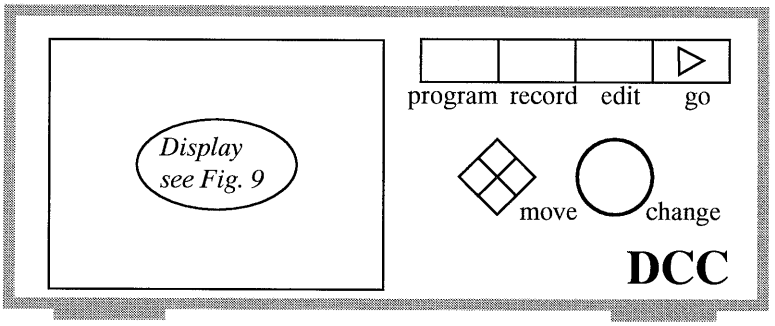


FIGURE 10. Impression of the front panel of the DCC prototype.
Indicated are only those buttons and knobs that are relevant to the recording task described in the text.

start the recording. The system now carries out the instructions on the record form, starting with the necessary tape-transport actions. During the recording the graphical representation is constantly updated. A new track emerges on the tape while the grey record bar gradually disappears.

3.4.2 LP description of the recording function

The interaction as described in the scenario is structured in three layers, each characterized by its own protocol. The protocols at the top and middle levels are shown in Figures 11 and 12. The protocol used at the bottom level is identical to the bottom-layer protocol of the DAT recorder (see Figure 8).

The top layer enables the selection of functions operating on the tape contents. This layer constitutes the highest level at which the system supports the users' intentions. Its two-step protocol includes the selection of the appropriate form, enables adjustment of the form and commitment to execution.
The primary top-level message that the user wants to make a recording is an interpretation of the bottom level message that the user has pressed the record button. The label 'record' should make users press the corresponding button when they have this intention. Therefore the label functions as E-feedback. After the button has been pressed, the record form is displayed on the screen. At this stage the record form functions as I-feedback, making users realize the system has interpreted they want to make a recording.
At this point users can adjust the form to their wishes. The middle layer supports the functionality for adjusting the contents of a form.
In the second step of the top-layer interaction the record form functions as E-feedback, since on the basis of this form users form expectations on what will happen after they commit to the system's interpretation by executing the physical action of pressing the go button. These expectations are strengthened by the additional E-feedback supplied in the graphical representation on the interpretation of the form. The label 'go' should make us-

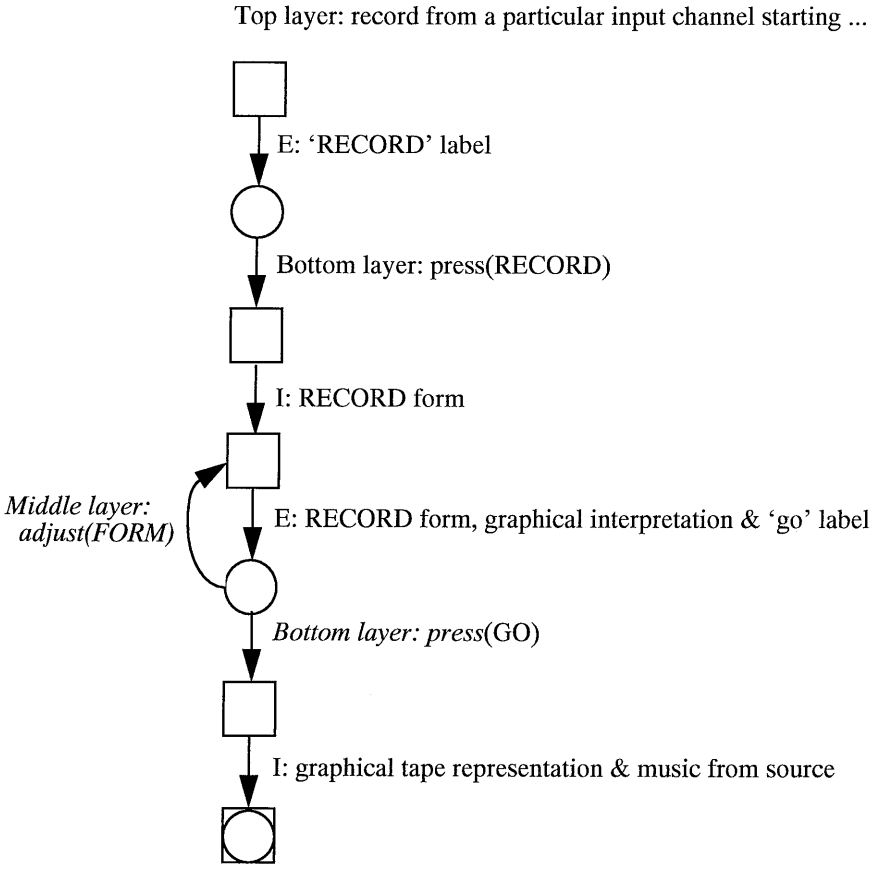


FIGURE 11. Top-layer interaction protocol for the recording function.

ers realize that they have to press that button to start the recording. The graphical representation, continuously updated according to the changing state of the tape in combination with the music, functions as I-feedback, making users realize the recording is in progress.

The middle layer is concerned with adjusting forms. This layer comprises two parallel protocols: one for selecting a field in the form and another for changing that field's contents.
A field is selected by positioning a frame around it. Since the position of the frame should make users realize whether the correct field has been selected the frame serves as I-feedback. The label 'move' near the cursor keys should make users realize that these keys enable repositioning of the frame. This label therefore functions as E-feedback.
The middle-layer message to change a field's contents is an interpretation of the bottom-layer message that the user has turned the change knob. The contents of a field serve as I-feedback, since they enable users to judge whether the field has been filled in correctly. In this protocol the frame functions as E-feedback, since its position influences the exact effect of turning the knob. The label 'change' should make users realize that the knob can be used for changing the contents of a field.

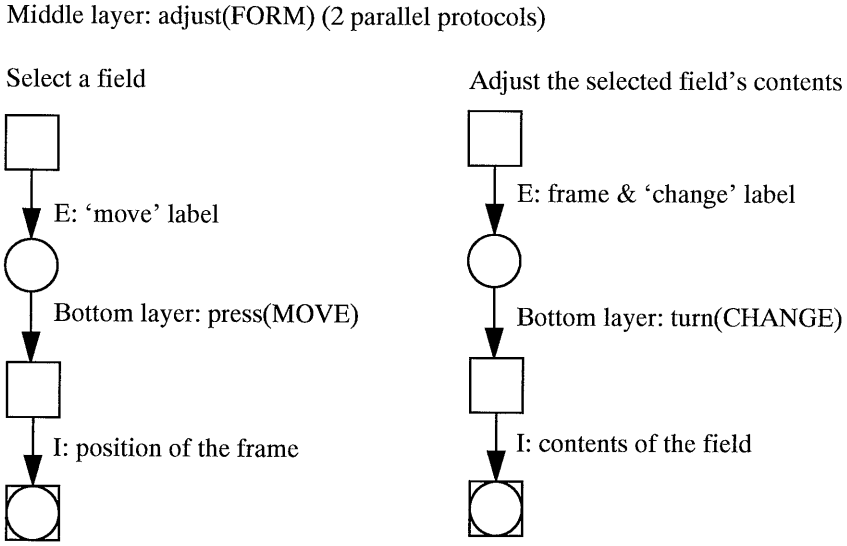


FIGURE 12. Middle-layer protocol for adjusting forms.

The bottom layer of the DCC prototype is organized in the same way as that of the DAT recorder (see section 3.3.2). The three layers provide independently encoded I-feedback of a different level of abstraction. The same information that serves as I-feedback at a particular level may also turn up as E-feedback in other interaction protocols when it indicates system settings relevant at a higher level.

3.4.3 Usability considerations

It will now be discussed to what extent the DCC prototype succeeded in following the guidelines derived in section 3.2.2, which should lead to certain improved usability aspects.

- *E-feedback*
  E-feedback is offered in several forms: the labels of the buttons on the front panel refer to the user intentions which are supported by the new interface. The forms show what system settings influence its interpretation. Finally, additional information in the tape representation reflects how the form will be interpreted. This E-feedback is expected to give rise to correct expectations on the part of the user as to how the system may behave. It helps users in selecting the correct buttons.
- *I-feedback*
  The I-feedback supplied is organized according to the layered interaction structure. By looking at the changes in the selected form and in the tape representation, the

effects of user messages can be immediately observed. This should make it easier for users to establish to what extent their intentions have been carried out. It is therefore expected that erroneous actions and system settings will be detected and adjusted at an early stage. Users will learn how to operate the interface faster.

- *Error and abort handling*
  Error and abort facilities are provided. The use of the rotary knob and the cursor keys for changing forms makes it intuitively clear to users how to obtain the inverse effect of a preceding modification: by turning the knob in the opposite direction or pressing the opposite cursor key. It is constantly possible to select another form.
- *Similarity of protocols*
  Thanks to the form-filling interaction protocol the more complex functions of the recorder are accessible in a similar way. This leads to increased internal consistency of the interface, which makes it easier to learn how to operate the interface.

In short, it is assumed that it will be easier to learn how to operate the DCC prototype because of its well structured E- and I-feedback and its high internal consistency.

3.5 Experimental comparison of the two user interfaces

The benefits and limitations of the LP model have to be assessed in a usability test: does the design based on the LP model yield an interface with improved usability? Therefore the usability of the newly designed DCC prototype was compared with that of the existing DAT recorder.

The usability of the recorders was measured according to ISO 9241 part 11 (1993, draft version). This standard uses the following definitions:

- usability is the effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments;
- effectiveness is the accuracy and completeness with which users achieve specified goals;
- efficiency is the accuracy and completeness of goals achieved in relation to resources expended;
- satisfaction is the comfort and acceptability of using a system.

The ISO standard gives guidance in specifying and measuring usability in the context of office work involving visual display terminals. The application area of consumer electronics differs from professional settings with respect to training, as users operating a digital audio recorder are not usually given formal instruction or coaching. Therefore, the recorder should aim at being self-explanatory: the functionality and the organization of the interaction have to become clear while users explore the system. That is why a fourth usability criterion is added in this paper: learnability, which is defined as the speed at which the user's effectiveness increases over time.

3.5.1 Experiment

The ISO standard indicates that, when measuring the usability of an overall system, a description is needed of the relevant characteristics of the context of use, which includes equipment, environment, users and users' task goals. These aspects will be described below.

Equipment and environment
The recorders were evaluated in a laboratory setting. The subjects were seated in front of a table with either the DAT recorder or the DCC prototype positioned at the centre. In addition, a simple CD player was available as a recording source. The DAT recorder is an existing product and was used unaltered. The newly designed DCC prototype was simulated on a SUN workstation. It was fully functional and very realistic. The prototype included auditory cues as winding sounds and a front panel with real push-buttons positioned in front of the computer screen.

Normally environmental factors may hamper information exchange between the user and the system at the physical level. For example, poor illumination may make the labels on the front panel hard to read or the position of the product may influence the accessibility of the buttons. In the experiment, care was taken to ensure that no environmental factors impeded information exchange at the physical level of communication.

Subjects
Both recorders were evaluated by subjects of the same user group. The subjects were between 20 and 40 years old and had Dutch as their native language. Their educational background was non-technical and of a vocational or university level. Subjects with such a profile fall within the target user group of digital audio recorders.

In total 20 subjects, 12 men and 8 women, participated in the experiment. They were divided into two groups of 10 subjects each, one for testing the DAT recorder and one for testing the DCC prototype. It was verified that the subjects working with the DAT recorder had on average the same experience with consumer appliances like VCRs, CD players and analogue cassette recorders as those working with the DCC prototype. The subjects had no experience with digital recorders.

Tasks
A set of 8 tasks was devised comprising five relatively simple tape-transport and playing tasks and three more complex tasks: a programming, a recording and a tape-editing task. That way the task set included common user goals. The order in which the tasks had to be executed was fixed. The subjects started with three tape-transport and playing tasks since users confronted with a new recorder are expected to explore this functionality first. After that, the subjects were confronted with the more complex tasks. In order to prevent the risk of subjects becoming discouraged when failing in these complex tasks, simple playing tasks were inserted between the programming and the recording task and between the recording and the tape-editing task. Since both recorders offer largely the same functionality, the same tasks were used for both recorders. These tasks were assumed to be repre-

sentative of the entire functionality offered by the recorders: if users were able to complete these tasks they had mastered all major aspects of the recorder's operation. The tasks were expressed in every-day wording. Technical terms were avoided as much as possible.

Experimental procedure
The experiment consisted of three sessions. In each session the same set of 8 tasks had to be completed. Before they started the first session, the subjects were given a written introduction of about 10 lines in which the most important aspects of the operation of the particular recorder being tested were indicated. They were allowed to study the introduction and explore the recorder for about 8 minutes. When they started the first session, the subjects were given the set of tasks and they were told a fully-fledged manual was available on request. The subjects were instructed to think aloud. If they had fruitlessly attempted to carry out a task for 20 minutes they were given the opportunity to proceed to the next task. At the end of the first session there was a 20-minute break before the second session. At the end of the second session the subjects were requested to fill in a questionnaire and they were explained the procedures for correctly executing the tasks. One week later the third session took place.

Measurements
For each task in each session several measurements were made. A task-completion score was objectively assessed by the experimenter on the basis of explicit criteria. If subjects successfully carried out a task they scored 1. That score was reduced for all aspects of the task erroneously executed. E.g. in the case of the recording task the completion score was decreased by 0.5 when users made the recording at the wrong position on the tape. The minimum completion score was 0. The time spent on each task was also measured. Furthermore, it was recorded whether the manual was requested or not. All sessions were video-taped to enable closer investigation of the subjects' behavior and identification of the usability problems encountered. These measurements made it possible to compare the interfaces in terms of effectiveness by comparing the task completion scores, in terms of efficiency by comparing the execution times and in terms of learnability by comparing how the completion scores and execution times evolved over the sessions. The degree of satisfaction was not directly assessed but was derived from the answers of the questionnaire.

3.5.2 Quantitative results

Effectiveness
An analysis of variance with repeated measures (MANOVA) was carried out on the task completion scores for the recorders, sessions and tasks. The main effects were found to be significant: recorder [$F(1,18)=6.3$, $p<.05$], session [$F(2,17)=18$, $p<.0001$] and task [$F(7,12)=5.8$, $p<.005$]. The task completion scores for the DCC prototype were on average higher than those for the DAT recorder. On average the scores increased over sessions. The significant main effect 'task' demonstrates that the subjects generally performed better in some tasks than in others. None of the interactions were found to be significant on a 5% level.

The average task completion scores for the two recorders per session are presented in Figure 13.

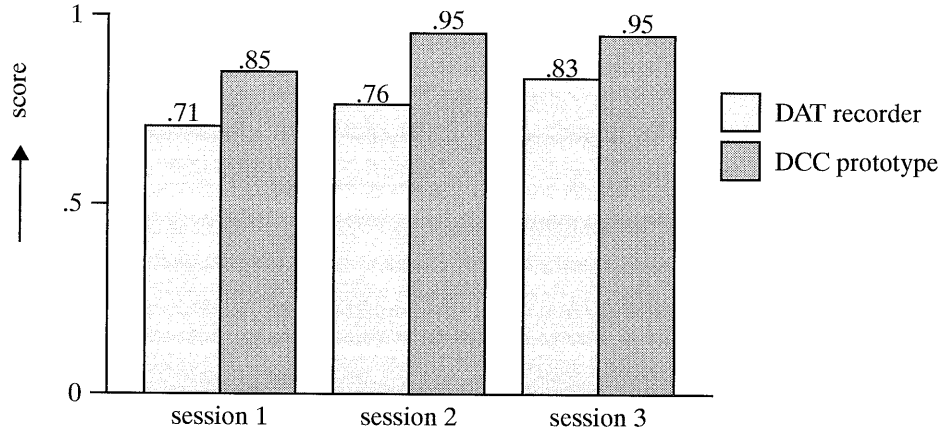


FIGURE 13. The average task completion scores per session for the two recorders.

A more detailed study of the task completion scores revealed that for sessions 1 and 2 the differences between the DAT recorder and the DCC prototype as shown in Figure 13 were significant [$F(1,18)=6.4$, $p<.05$ and $F(1,18)=7.1$, $p<.05$]. For session 3 this was no longer the case on a 5% level. For the DAT recorder the increases between sessions 1 and 2 and between sessions 2 and 3 were significant [$F(1,36)=4.0$, $p=.05$ and $F(1,36)=5.9$, $p<.05$]. For the DCC prototype the increase between sessions 1 and 2 was significant [$F(1,36)=13$, $p<.001$]. There was no significant difference between the scores of sessions 2 and 3.

Efficiency

An analysis of variance with repeated measures (MANOVA) was carried out on the task completion times for the recorders, sessions and tasks. All main effects were found to be significant: recorder [$F(1,18)=15$, $p<.001$], session [$F(2,17)=57$, $p<.0001$] and task [$F(7,12)=63$, $p<.0001$]. The task completion times for the DCC prototype were on average lower than those for the DAT recorder. On average the first session took longer than the other two sessions. The significance of the main effect 'task' demonstrates that the subjects generally performed better in some tasks than in others. The only significant interaction was that between the tasks and recorders [$F(7,12)=7.9$, $p<.001$]. The other interactions were not significant on a 5% level.

The average completion times required by the subjects for the total set of tasks are presented in Figure 14 for the two recorders.

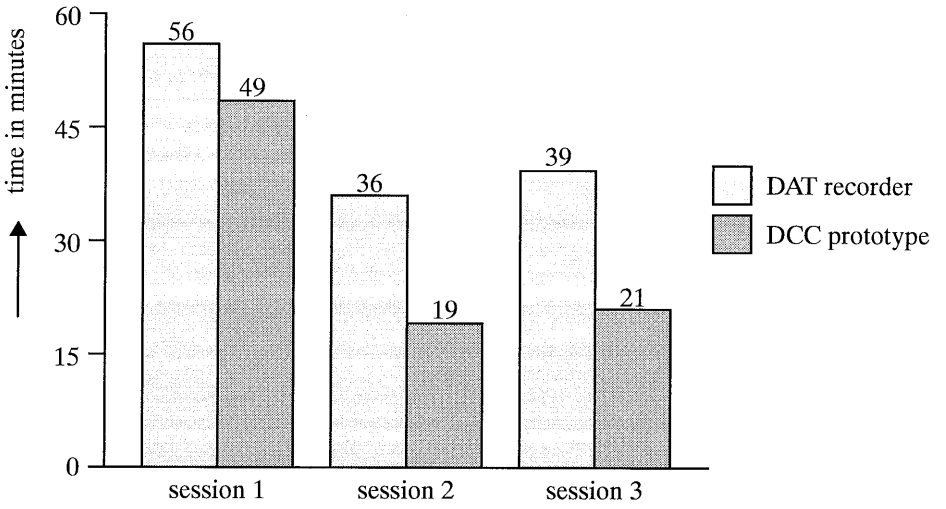


FIGURE 14. The average amount of time needed to complete the total set of tasks per session for the two recorders.

A more detailed study of the task completion times revealed that for sessions 2 and 3 the differences between the DAT recorder and the DCC prototype as shown in Figure 14 were significant [$F(1,18)=15$, $p<.001$ and $F(1,18)=11$, $p<.005$]. However, for session 1 this was not the case on a 5% level. For both the DAT recorder and the DCC prototype the decrease between sessions 1 and 2 is significant [$F(1,36)=28$, $p<.0001$ and $F(1,36)= 62$, $p<.0001$]. The difference between sessions 2 and 3 is not significant.

Satisfaction

In general, the subjects who had used the DCC prototype expressed a more positive attitude in the questionnaire than the subjects who had used the DAT recorder. This was most apparent when the subjects were asked what they thought of the experiment. In the case of the DAT recorder eight subjects said that they found the experiment difficult whereas two subjects said they liked the experiment. In the case of the DCC prototype only two subjects had found the experiment difficult, whereas four subjects said they had liked the experiment and another four subjects indicated that they had ultimately appreciated the experiment although they had experienced difficulties at first.

Though this information does not indicate satisfaction in a direct way, it does reflect a difference between the two recorders in terms of some subjective aspect of usability.

Learnability

In the case of both recorders a learning effect was apparent because both the scores and the task completion times yielded a significant effect over the sessions. A significant effect over the recorders was also observed: the task completion scores and the task completion times were generally better for the DCC prototype than the DAT recorder. Since

the subjects were unfamiliar with the particular recorder tested and digital recorders in general this indicates that subjects learned how to operate the DCC prototype faster than the DAT recorder. The following more detailed evidence supports this:

- The average completion score of session 1 is significantly higher for the DCC prototype than for the DAT recorder (see Figure 13).
- In session 2 the average completion score was .95 for the DCC prototype. This high score was again obtained in session 3. This indicates that the subjects had already mastered the operation of the interface in session 2 and only made minor mistakes after session 1. On the other hand, the completion scores of the subjects who operated the DAT recorder significantly increased by .07 from session 2 to session 3.
- In session 1 the average task completion time was not significantly shorter for the DCC prototype than for the DAT recorder. However, the average task completion times for sessions 2 and 3 differ significantly: the average task completion time for the DCC prototype is about half that of the DAT recorder (see Figure 14).
- The data regarding manual use constitute further evidence (see Figure 15). An analysis of variance with repeated measures (MANOVA) of these data revealed a significant difference between the two recorders [$F(1,18)=29$, $p<.0001$]. This indicates that the subjects who operated the DAT recorder generally consulted the manual more often.

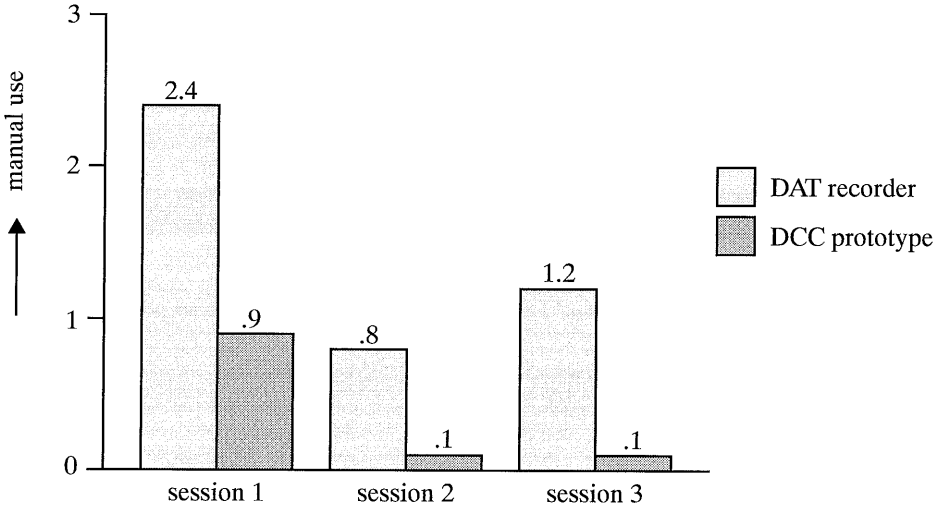


FIGURE 15. The average total number of times that the subjects had to consult the manual to be able to complete all the tasks per session for the two recorders.

Figure 16 summarizes the data presented in Figures 13 and 14 and shows the average task completion scores as a function of the cumulative total task completion times. This yields a comprehensive picture indicating how quickly effectiveness increases over time. The difference in learnability is apparent.

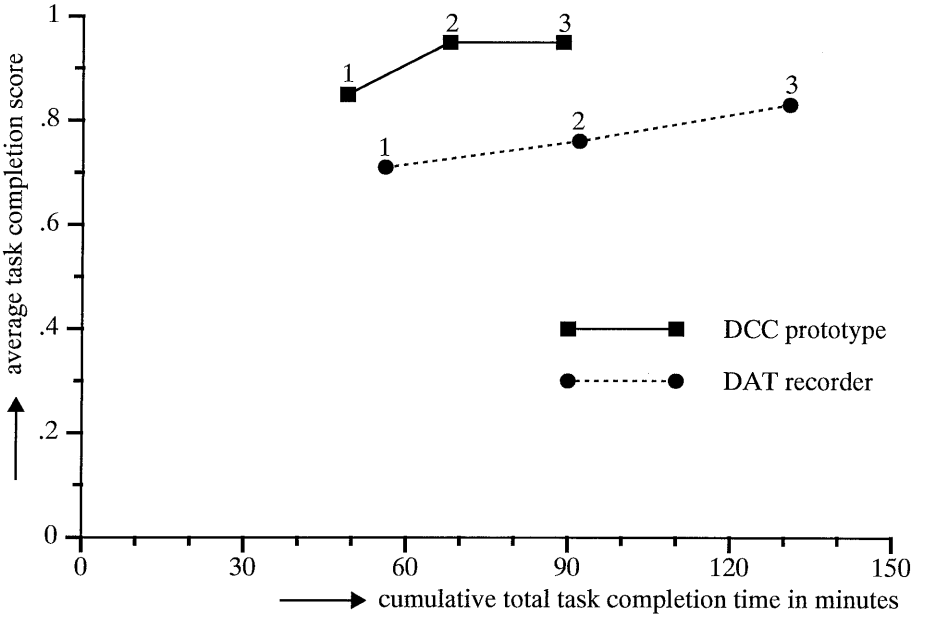


FIGURE 16. The 'learning curves' for the two recorders. For each session a data point is plotted, representing the average task completion score for that session on the vertical axis, and the total time spent in interaction with the recorder (cumulative total task completion times) on the horizontal axis. The graphs thus give an impression of how much was learned in what amount of time.

3.5.3 Observational results

Usability
The data presented above show that the DCC prototype is more usable than the DAT recorder in terms of effectiveness, efficiency, satisfaction and learnability. Since the environment, the functionality and the set of tasks were kept constant in both evaluations, and because the subjects were comparable in terms of age, educational background and experience with consumer appliances, the measured usability differences between the two recorders must be mainly attributable to the fact that they have different user interfaces.

These two user interfaces differ with respect to the extent that they are in line with the LP model: the design of the user interface of the DCC prototype was based on the LP model whereas the DAT user interface deviates from the LP model in various respects. It remains to be discussed to what extent the user interface aspects indicated by the LP model account for the usability differences between the two interfaces. By reviewing the video tapes it was investigated whether the expected usability consequences had actually occurred and whether any unexpected usability problems had turned up.

Observed usability aspects addressed by LP
The discussion of what elements of the LP model account for the usability differences between the DAT recorder and the DCC prototype will be based on the derived guidelines: how they were expected to work and how they actually worked (see sections 3.2.2, 3.3.3 and 3.4.3).

- *E-feedback*
  The subjects who used the DAT recorder often had trouble deciding which buttons had to be pressed for the various tasks because many labels were rather cryptic and did not give rise to expectations as to the effect of pressing these buttons. The buttons involved in tape editing presented the most problems. The subjects were also uncertain about the relevant settings and their influence on the ultimate effect. The DAT recorder provides virtually no E-feedback of this kind.
  The DCC prototype also had labels that were not immediately understood: viz. the labels of the buttons involved in changing the contents of a form. The subjects understood the relevance of the fields on a form quite well and had no trouble in predicting the effect that the execution of a form would have on the tape. The additional information in the graphical tape representation was of help in interpreting the forms and making the predictions.
  Thus, E-feedback proved to have a strong influence on usability. However, the mere availability of E-feedback at each level is not sufficient. It must also be of high quality: it must provide users with sufficient information for predicting the effect of a message.

- *I-feedback*
  Although the subjects who used the DAT recorder could often infer from lower-level I-feedback that the recorder was doing something with the tape, they had problems determining the effects of their actions because I-feedback was lacking at the highest level. For example, when some buttons were pressed the corresponding label was repeated in the display. Although these subjects realized that the system had reacted, they were often unable to infer the effects from what they knew about the previous state of the recorder in combination with their knowledge of the function of the button. So the subjects had to establish the effects of their actions by transporting and playing the tape. As a result, they found it difficult to determine what action had caused the effect.
  The subjects who used the DCC prototype were quite capable of immediately assessing the effects of their messages by observing the changes on the display.
  This leads to the same conclusion as for E-feedback, namely that the mere availability of immediate I-feedback at each level is not sufficient. It must also be of high quality: the users must be provided with sufficient information for determining to what extent the intentions have been carried out, for detecting communication failures and for learning the operation of the interface by linking their actions to system responses.

- *Error and abort handling*
  No usability differences due to lacking error and abort facilities were observed. The DAT recorder and the DCC prototype did not differ much in this respect. All the subjects frequently used these facilities in exploring the interface. This indicates that these facilities form an indispensable part of interaction protocols.
- *Similarity of protocols*
  When confronted with the form-filling strategy for the first time, the DCC subjects required some time to grasp this protocol. When using it for the second time, for a different task, they made extensive use of the experience gained during the first encounter. The subjects were quite capable of transferring their experience to another task and generalizing the interaction protocols. During the first session they often needed to consult the manual only once: namely for the programming task when they were confronted with the form-filling strategy for the first time. The DAT recorder shows virtually no internal consistency of this kind. During the first session the subjects often needed to consult the manual up to three times: namely for the programming, the recording and the tape-editing tasks (see Figure 15). With the DCC prototype these tasks could all be executed using the same form-filling strategy whereas the DAT recorder employed three different interaction protocols. It is therefore concluded that similarity of protocols is indeed an important asset predicted by the LP model.

To summarize, it can be observed that the guidelines derived from the LP model have worked as assumed but that the influence of the quality of the E- and I-feedback has been underestimated.

Observed usability aspects not addressed by LP
Although the LP model provides a unified and consistent framework for structuring user-system interaction, it may very well be incomplete and may miss some aspects that also affect usability. A usability issue that is not addressed in the LP model but whose importance was observed in the video tapes is the familiarity of subjects with the user interface concepts employed. Three observations lead to this conclusion:

- In general the subjects had little trouble executing the tasks involving playing and transporting the tape, despite the low quality of the E- and I-feedback. This can be explained by their familiarity with CD players and analog cassette recorders. This experience could be easily transferred to the DAT recorder and the DCC prototype because the same interaction concepts were applied.
- The facts that the subjects were not familiar with the user interface concepts of the DAT recorder and that they differed from their intuitive notions caused confusion. The subjects' intuitive notion was that a track is the central concept that the various functions work on. This is probably due to their familiarity with CDs. However, the DAT user interface uses the ID concept: e.g. a 'start ID' marking the tape position at which a track begins can be 'written' on the tape and can be 'erased'. This led to dis-

crepancies between the subjects' expectations and the actual behavior of the system. Sometimes the subjects confused their intuitive notions with the recorder's concepts: e.g. they feared losing the music of a track when erasing its start ID.

- The subjects who used the DCC prototype had problems learning to fill in a form. Although all the subjects succeeded in mastering the form-filling protocols, these protocols caused serious delays in the task completion times of the first session. Because the subjects were unfamiliar with this kind of form-filling, it took them a long time to learn these new interaction concepts.

To summarize, it can be observed that the users' familiarity with the user interface concepts employed also has an effect on usability. Since this factor was found to be influential in the case of both interfaces, it is unlikely that it fully accounts for the measured usability difference between the DAT recorder and the DCC prototype.

3.6 Discussion

The goal of this paper was to assess the benefits and limitations of the Layered Protocols model for the analysis and design of user interfaces in the domain of consumer electronics. A first step towards the achievement of this goal was the derivation of a set of guidelines from the LP model. These guidelines were used to analyze an existing interface and to design a new one. Next, the hypothesis was formulated that the user interface based on the LP guidelines is more usable, which was investigated by measuring the usability of the two interfaces. Finally it was discussed to what extent the differences in usability can be explained by deviations of the interfaces from these guidelines. This assessment of the LP model will be discussed in this section.

Describing the user interfaces in LP terminology
Analyzing and designing user interfaces with the aid of the LP model involves describing the interface in terms of the model. Although the LP model is so general that it can describe virtually any interaction, the amount of work involved proved to be different for the two interfaces. Describing the DCC prototype was easy since it was explicitly designed according to the LP model. But the analysis of the DAT recorder was rather tedious because each function had to be described separately. The interaction protocols for different functions showed little similarity. Since the feedback supplied does not reflect a layered structure, it was questionable whether an LP analysis would be appropriate. However, the user's part of the interaction shows a layered structure: the execution of a function like record means that some settings, such as the tape head position, have to be adjusted. The lack of layered feedback made the analysis of the DAT recorder more difficult. It is also assumed that this lack makes it harder for users to become aware of the layered interaction structure.

Guidelines
The LP model as presented by Taylor (1988a) is a very general interaction model. For this study, a selection was made of the interaction concepts introduced in the LP model. The most important concepts were elaborated into guidelines indicating potential usability consequences. By making the model operational in this way it becomes possible to test whether the model and the guidelines correctly indicate factors influencing usability.
On the basis of the interface descriptions in combination with the guidelines, potential usability problems were pointed out. Although the usability issues indicated by the guidelines actually occurred, it was felt that it should be possible to pinpoint usability consequences more precisely on the basis of the detailed interface descriptions. The current guidelines are not specific enough in relation to the detailed LP descriptions. More hands-on experience is needed to extend and refine the guidelines.

Evaluating usability
The ISO 2941 standard was used to compare the usability of the two interfaces. Because the assessment was carried out in the domain of consumer electronics, learnability was added as a usability criterion. It was defined as the speed at which the users' effectiveness increases with time. Though it can be argued that learnability is simply the way in which usability develops with time, it is considered important enough to be treated as a separate usability criterion. Learnability is more likely to influence what functionality will actually be used on the long term than efficiency. Nielsen (1993) even considers it the most fundamental usability attribute. Unfortunately the ISO 2941 standard does not address this issue.
Because the experiment was conducted in a laboratory setting, it has to be considered whether the measured usability relates to the usability experienced in real life. A major difference between the laboratory setting and a real-life situation concerns the goals: the subjects were given explicit, well-defined tasks whereas intentions of users may be less articulate. It is also felt that the subjects were more tenacious in their attempts to fulfil a task than users would be in real life. Nevertheless, the experimental conditions in this study make it plausible that the measured usability differences reflect real-life usability differences.
While usability focuses on making the system's functionality available to users, the usefulness of a system indicates whether its functionality serves a purpose for its users. In real life, a good match between the system's functionality and the users' intentions is likely to yield a more valuable system for its users. That is why in this study the two recorders were kept functionally equivalent.

The LP model
A general interaction model should be a useful tool for designing better conceived and more usable interfaces. In principle, it should assist in structuring the interaction and explaining the role of each piece of information exchanged while indicating usability consequences. A general interaction model cannot be expected to indicate the usefulness of functionality, nor can it be expected to describe the actual contents of interaction messages. However, it could indicate criteria for high-quality feedback and provide simple usability tests for measuring quality differences between alternative messages and interaction structures. In this way an interaction model can decrease the need for more extensive user

tests in interface design and help in reaching a basic level of usability.
This study has shown that the LP model is a step in the right direction. Its distinction between E- and I-feedback indicates the different roles feedback can have. It advocates structuring interaction in a layered way while the guidelines indicate potential usability consequences. The conclusion of this paper is that these interaction aspects actually influence the usability of a system.
The main drawback of the LP model is its lack of precision. More explicit criteria for distinguishing between layers and classifying information as E- or I-feedback should make the LP model more clear-cut. The guidelines must also be refined to predict usability consequences more precisely. The LP model would be far more useful in user-interface design if simple standardized user tests were available for evaluating interaction per layer, while the overall interaction architecture would guarantee a basic level of usability.

3.7 Conclusion

On the whole, it can be concluded that the Layered Protocols model indicates interaction aspects influencing the usability of user interfaces. Guidelines can be derived from the LP model that can be of help in reaching a basic level of usability in user interface design.

Some of the most important aspects of the LP model include the layered organization of the interaction in combination with immediate high-level I-feedback, which facilitates early detection of communication failures. Furthermore, E-feedback guides users in coding their intentions. The quality of E- and I-feedback turned out to be crucial: E-feedback must provide users with sufficient information for predicting the effect of a message, and I-feedback must provide users with sufficient information for determining to what extent their intentions have been carried out and for detecting communication failures.

The LP model also indicates that the organization of I-feedback at the various levels strongly influences the learnability of an interface. It helps users to establish the effects of their messages immediately. A second factor contributing to learnability is an optimum similarity between interaction protocols, which leads to increased internal consistency within user interfaces. In consumer electronics in particular high learnability is essential, because most consumers will have had no formal instruction or coaching.

One influential usability factor for which the LP model does not account was observed: the users' familiarity with the interaction concepts applied in the user interface. With this limitation in mind, the overall conclusion of this assessment is that the Layered Protocols model indeed addresses user interface aspects influencing usability and can thus be of help in reaching a basic level of usability in user interface design.

We gratefully acknowledge the layered feedback received from Don Bouwhuis, Frits Engel, Floris van Nes, Martin Taylor and John de Vet on earlier versions of this paper. We thank Rudy van Hoe and René Wijnen for their help with the data analysis.

3.8 References

Card, S.K., Moran, T.P. & Newell, A. (1983). *The Psychology of Human-Computer Interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Engel, F.L. & Haakma, R. (1993). Expectations and feedback in user-system communication. *International Journal of Man-Machine Studies,* **39**, 427-452.

HUSAT Research Centre (1988). Human Factors Guidelines for the design of Computer-Based Systems, Loughborough.

ISO (1993). Guidance on specifying and measuring usability, ISO CD 9241 part 11 (draft).

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

Marshall, C., Nelson, C. & Gardiner, M.M. (1987). Design Guidelines. In: M.M. Gardiner & B. Christie (Eds.) *Applying Cognitive Psychology to User-Interface Design*, pp. 221-278. Chichester: Wiley.

Nielsen, J. (1993). *Usability Engineering.* Boston, MA: Academic Press.

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

Payne, S.J. & Green, T.R.G. (1986). Task-action grammars: a model of the mental representation of task languages. *Human-Computer Interaction,* **2**, 93-133.

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

Taylor, M.M. (1988b). Layered protocols for computer-human dialogue II: Some practical issues. *International Journal on Man-Machine Studies,* **28**, 219-257.

Taylor, M.M. (1992a). *Principles for Intelligent Human-Computer Interaction, a tutorial on Layered Protocol Theory.* DCIEM report no. 93-32, North York, Ontario, Canada.

Interlude

We are now half way through this thesis. The previous two chapters described a first step towards finding ways in which the Layered Protocols framework can provide guidance in the design of easy-to-use user interfaces for novice and occasional users:

- chapter 2 studied the concepts introduced by the Layered Protocols framework and elaborated upon the role of expectations in user-system interaction leading to the distinction between E-feedback and I-feedback;
- the first part of chapter 3 was devoted to operationalization of these LP concepts and their use in the design of a full-scale user-interface for DCC recorders to be used by novice and occasional users;
- in the second part of chapter 3 the usability of that user interface was compared with that of the user interface of an existing digital audio recorder which was only partly in line with the LP model. The newly designed user interface was found to be more usable than the existing one.

In the following three chapters a next step will be taken:

- chapter 4 is based on chapter 2. It will study the role of E- and I-feedback in greater detail with reference to the Perceptual Control Theory. It will be indicated how E- and I-feedback at the different interaction levels can create perceptions that induce users to build up correct intention hierarchies;
- chapter 5 will provide a very detailed description of a second user interface for DCC recorders. That description will be more systematic than those presented in chapter 3. It will use the findings of chapter 5 in that the role of each piece of information that the interface provides to users will be indicated;
- in chapter 6 the usability of this second user interface will be determined and compared with the usability measurements presented in chapter 3.

Chapter 4

Towards explaining the behavior of novice users[1]

Abstract

Novice users confronted with a new piece of consumer equipment often start using it right away. Some user interfaces allow users to do this successfully. This chapter tries to explain why by analyzing four simple user interfaces in the framework of the Perceptual Control Theory (Powers, 1973). It is described how the interaction with novice users is assumed to evolve. An explanation is provided why novice users are expected to select the appropriate physical actions and execute them in the correct order.

The analyses point out how E- and I-feedback (chapter 2: Engel and Haakma, 1993) at the different interaction levels should create perceptions that induce users to build up correct intention hierarchies. It is made plausible that label following, difference reduction and analogical reasoning are likely to be among the psychological mechanisms people use to link intentions to sub-intentions. Among the mechanisms for ordering sub-intentions are selective feedback presentation, difference reduction and forward and backward chaining.

4.1 Introduction

Novice users confronted with a new piece of consumer equipment often start using it right away. They figure out how the product works while using it. For some products this strategy seems to work, for others it does not (chapter 3: Eggen, Haakma & Westerink, 1996). This chapter attempts to shed some light on the reason why: why do novice users do what they do?

This chapter analyses four simple user interfaces. The four could all be parts of user interfaces for CD-R players, compact disc players that can play music as well as record it (the R stands for recordable). An account will be given of how the interaction between

1. A slightly modified version of this chapter has been submitted for publication as a paper by Haakma, R. with the title 'Why do novice users do what they do' in the International Journal of Human-Computer Studies.

product and novice user is assumed to evolve on the basis of the Perceptual Control Theory (Powers, 1973; Taylor, 1992). The interaction elements that should make users select the appropriate actions and execute them in the correct order will be indicated.

4.2 Stages and levels in Perceptual Control Theory

4.2.1 Stages

Perceptual Control Theory (PCT) states that all behavior is aimed at controlling perceptions. Within the context of user-system interaction this paradigm can be formulated as follows: the users' physical actions are targeted towards controlling their perceptions by controlling the system's feedback. This approach is reflected in the perception-control loop used in this chapter (see Figure 1). In compliance with PCT, four stages of user activity, named after Norman (1984), are identified. First, users are expected to develop some idea of what they wish to perceive (*intention-formation stage*). When not actually perceiving what they want to perceive, users may decide to interact with the system and try to realize the desired perceptions (*evaluation stage*). During the *action-selection stage* users decide how to interact with the system. The selected actions are expected to reduce the discrepancy between what users currently perceive and what they want to perceive. During the last stage, users carry out the actions they selected (*action execution*). This is the moment at which users actually give input to the system (black arrow from user to system in Figure 1). To users, the system is a kind of black box hiding its internal workings. The black arrow pointing from system to user indicates the information generated by the system and perceived by users. This is what the interaction is all about. In compliance with the Layered Protocols Theory (Taylor, 1988; 1992), this information is called *I-feedback*. The I stands for 'interpretation': this feedback reflects the system's interpretation of the user's actions. In addition, the system is assumed to provide users with information guiding them through the intention-formation and action-selection stages (the two grey arrows). This information is called *E-feedback* (chapter 2: Engel and Haakma, 1993). The E stands for 'expectation': it should raise expectations with users as to the effect of potential actions.

The stages in user-system interaction are illustrated by the first interface example. The interface allows users to play a CD starting at the track of their choice[1]. Figure 2 shows the interaction means: 99 buttons (because a CD can have up to 99 tracks) and a two-digit display. The display shows the number of the track currently being played. Users can listen to another track by pressing the corresponding button. For instance, after a user has pressed the button labeled '12', the player will start playing the 12th track of the CD currently loaded.

1. To keep the interface example simple, no functionality has been introduced to stop the music, load and unload a CD and switch the player on or off.

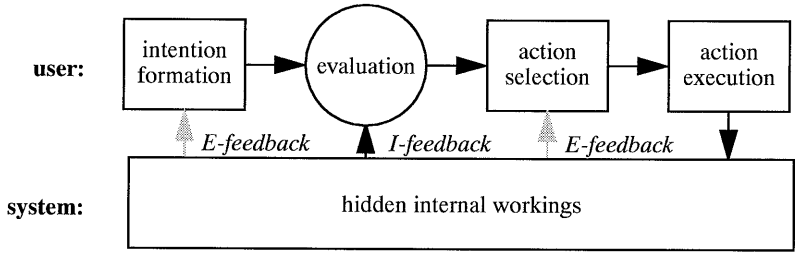


FIGURE 1. The perception-control loop.
The four stages are shown at the user's side of the loop. The system is shown as a kind of black box hiding its internal workings. Users can only observe the E- and I-feedback provided by the system. By executing actions, like pressing buttons, users can make the system change its I-feedback.

Suppose users are listening to a CD but do not like track 3 that is currently being played. According to PCT, the interaction between the user and the system is now assumed to evolve as follows:

- **Intention formation**
  The track list on the CD box gives an overview of the music on the disc by naming and numbering the tracks. This E-feedback guides users in forming intentions like 'I want to listen to track 12'; intentions that match with what the player can do.
- **Evaluation**
  Users will realize that their intention has not been satisfied because they hear the wrong music and/or see an incorrect track number in the display (I-feedback). This discrepancy urges users to take actions to reduce the difference between the actual and the intended perception.
- **Action selection**
  The buttons on the front panel of the player provide users with an overview of possible actions. The labels on the buttons assist users in selecting the right one (E-feedback). The label '12' suggests to users to select the corresponding button for playing the 12th track. Polson and Lewis (1990) call this selection mechanism the *label-following heuristic*: 'Essentially, learners determine which action appears to lead most quickly to the goal by comparing the description of the available actions with the description of the goal'.
- **Action execution**
  It should now be trivial for users to execute the selected action, i.e. to press button '12', though problems may occur when the buttons are small or the CD player is hard to reach.

When they see '12' appear in the display and hear the corresponding music (I-feedback) users will realize that their actions have taken effect. The actual perception has changed and now matches the intended perception. There is no reason for further interaction until users change their mind (i.e. change their intention) or the music ends.

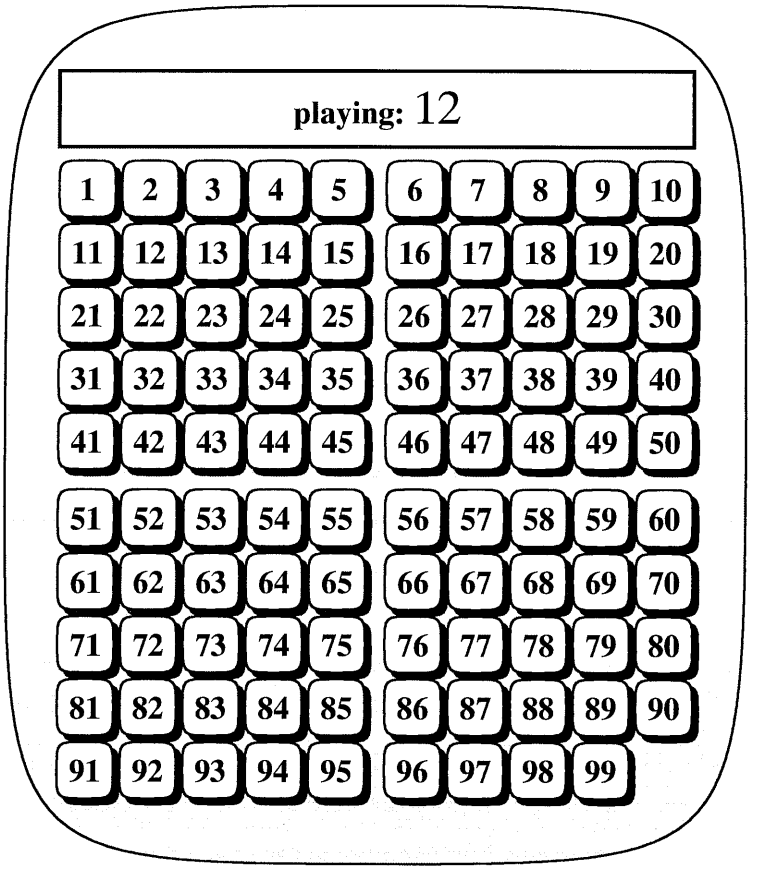


FIGURE 2. A user interface for playing music from a CD. The labels refer to the track numbers indicated on the CD box. The display indicates the track currently being played.

4.2.2 Levels

When we look more closely, we note that the interaction actually involves two hierarchically-related control loops. The top-level control loop is about controlling the music as indicated above. The lower-level control loop involves pressing buttons: the appearance of the buttons on the front panel (E-feedback) assists users in forming the (sub-) intentions of pressing those buttons. After a button has been pressed, kinesthetic feedback in combination with the click of the button (I-feedback) should make users realize that the intention of pressing a button has been achieved (evaluation). The action-selection stage is about how to get the button pressed: users may use a finger or a pen or may ask someone else to do it.

Users will have to link the two levels of interaction. This is enabled by the labels on the buttons. The label '12' is linked to track number 12 (top level) and the position of that label links it to the appropriate button (lower level). Or, stated otherwise, the location of the label '12' makes pressing the associated button a relevant sub-intention for achieving the overall intention of listening to track 12. The label-following heuristic explains why users would find this sub-intention relevant indeed within the context of the higher-level intention.

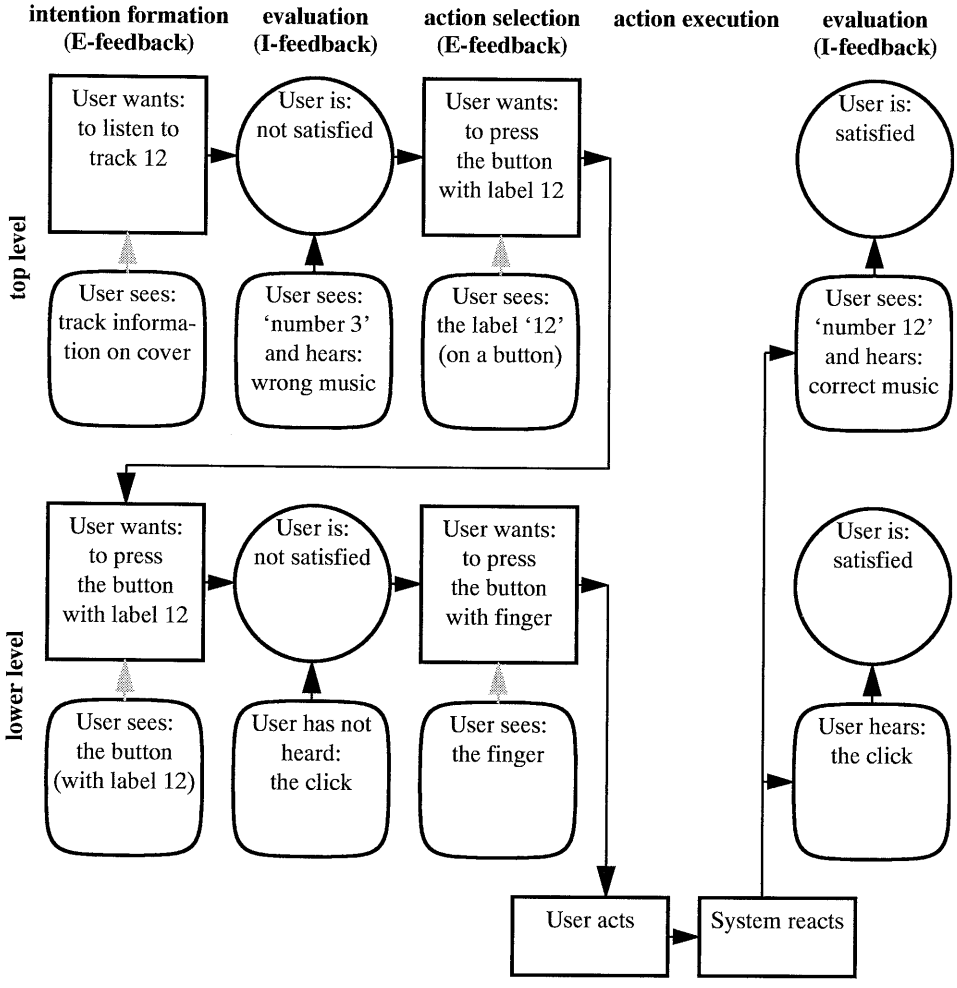


FIGURE 3. A full overview of how the interaction is assumed to evolve.

Figure 3 shows in full how the interaction is assumed to evolve. The square boxes represent the stages being processed. The rounded boxes indicate the user perceptions generated by the system's E- or I-feedback to support the different stages. The square boxes in the top row represent the stages involved in the top-level perceptual-control loop. The

square boxes in the second row belong to the lower-level control loop. At the bottom the physical interaction with the system takes place.
Figure 3 shows that the top-level action-selection stage and the lower-level intention-formation stage are tightly linked. The E-feedback, which is the same for the two stages, comprises a top-level ('12') as well as a lower-level element (the button). This indicates that action selection can be thought of as intention formation one level down.
Figure 4 is a condensed version of Figure 3 focussing on the intentions to be formed at the different levels and the E- and I-feedback facilitating the formation of this intention hierarchy.

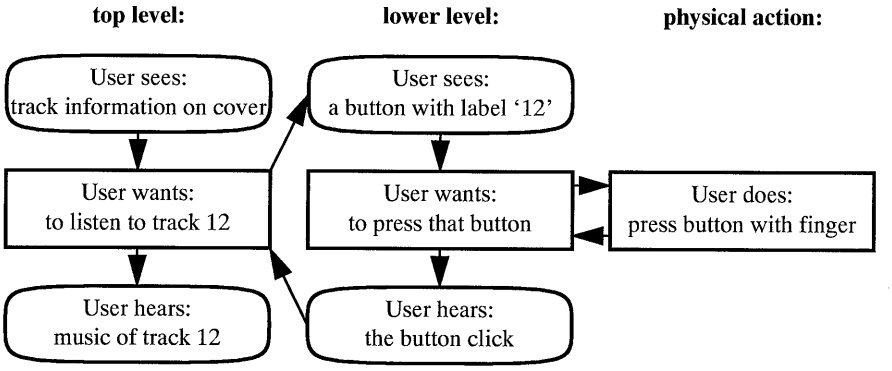


FIGURE 4. The intention hierarchy of the first user interface.
The left column relates to the top-level control loop, the middle column to the lower-level control loop. To the right the actual interaction with the system is shown. The square boxes indicate the intentions users are assumed to form. The rounded boxes at the top indicate the information provided to assist in intention formation/action selection (E-feedback). The rounded boxes at the bottom show the user perceptions to be controlled (I-feedback).

4.3 Difference reduction

The first user interface has a large number of buttons. They take up a lot of space while the higher-numbered buttons will hardly ever be used because most CDs do not have that many tracks. The user interface shown in Figure 5 requires only two buttons. It nevertheless offers the same functionality. The 'next number' and 'previous number' buttons increase and decrease the track number in the display, respectively. The player plays the indicated track.[1]

1. Again, no functionality has been introduced to stop the music, load and unload a CD and switch the player on or off. The player automatically starts playing the first track after a CD has been loaded.

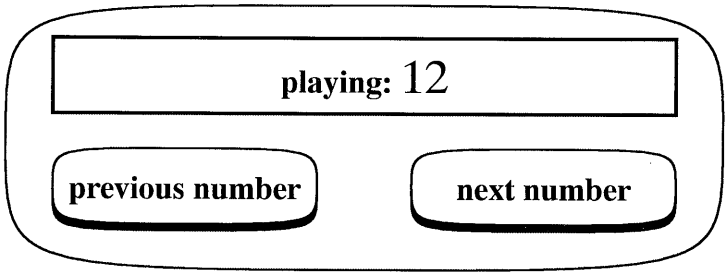


FIGURE 5. The second user interface. It offers the same listening functionality as the first one. The 'previous number' and 'next number' buttons change the track number in the display and thus the track being played.

From a PCT point of view there are two issues at stake. The first one is about action selection. When, for example, users want to listen to track 12 and observe 'playing: 3' in the display why would they press the 'next number' button? This behavior cannot be explained by the label-following heuristic. The second issue is about evaluation. Why would users press the 'next number' button another eight times while the first time did not induce the intended perceptions?
Of course, users select the 'next number' button and keep pressing it because they are closing in on their goal: the 4th track is closer to track 12 than track number 3. Reaching track 4 is experienced as a successful step in the right direction. The intention hierarchy is shown in Figure 6.

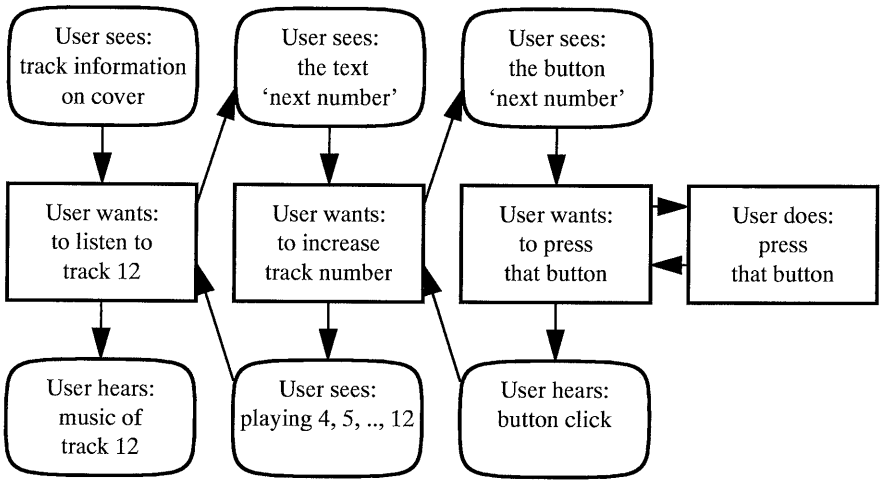


FIGURE 6. The intention hierarchy of the second user interface.

The numbering of the tracks on the CD box in combination with the labels 'next number' and 'previous number' gives rise to a sense of proximity or distance. Users select actions they consider to be a step in the right direction. Polson and Lewis (1990) call this *hill-climbing* behavior. In PCT it is called *difference reduction* (see evaluation stage in 4.2.1).

By using the difference-reduction strategy the number of buttons can be largely reduced. However, users may have to press the 'next number' or 'previous number' buttons many times before they reach the intended track. This makes the interaction tedious for users. That is why many commercially available CD players also offer a numeric keypad for entering track numbers.

4.4 Two-step interaction

Most commercial CD players do not start playing right away. They normally remain idle until the 'play' button is pressed. This means that users do not have to listen to (the first parts of) unwanted tracks. Another advantage is that the start of the music is better under control. This is useful for example when recording a disc to tape. Figure 7 shows such an interface[1].

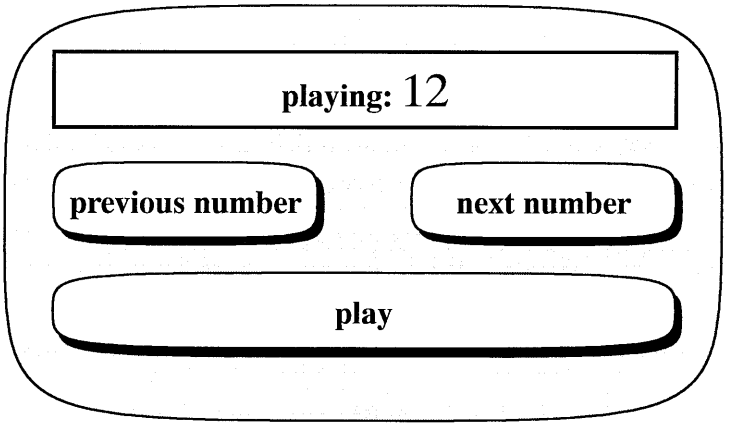


FIGURE 7. The third user interface. The 'previous number' and 'next number' buttons manipulate the track number in the display. After the 'play' button has been pressed the player will show 'playing' in the display and start playing the indicated track.

This interface induces the formation of two sub-intentions: adjusting the track number in the display and getting the player to start playing. The track number in the display and the word 'play' on the lower button are both associated with the top-level intention of 'hearing the music of track 12'. The player shows both at the same time.

1. Again, no functionality has been introduced to stop the music, load and unload a CD and switch the player on or off.

When users start focussing on the track number they are expected to show the hill-climbing behavior at first, as described in the previous section. When the correct track number shows up in the display, the track is still not being played. Since the 'next number' and 'previous number' buttons will no longer be of any help in getting closer to the goal, users are now expected to be triggered by the 'play' label; which should induce them to press the associated button. This line of reasoning is called *forward chaining*.
The line of reasoning is slightly different when users start focussing on the 'play' label. Users are expected to notice, maybe by trial and error, that when they press the 'play' button the player will start playing the track indicated in the display. This should make them change the track number first. This line of reasoning is called *backward chaining* (see e.g. VanLehn, 1989).

In both lines of reasoning the critical issue is whether users realize that the system's reaction to pressing the play button depends on the track number in the display. The user interface of Figure 7 indicates this dependency implicitly. The intention 'listen to track 12' in combination with the word 'play' should make users wonder 'what will be played?' and direct their attention to the track number in the display. The intention hierarchy is shown in Figure 8.

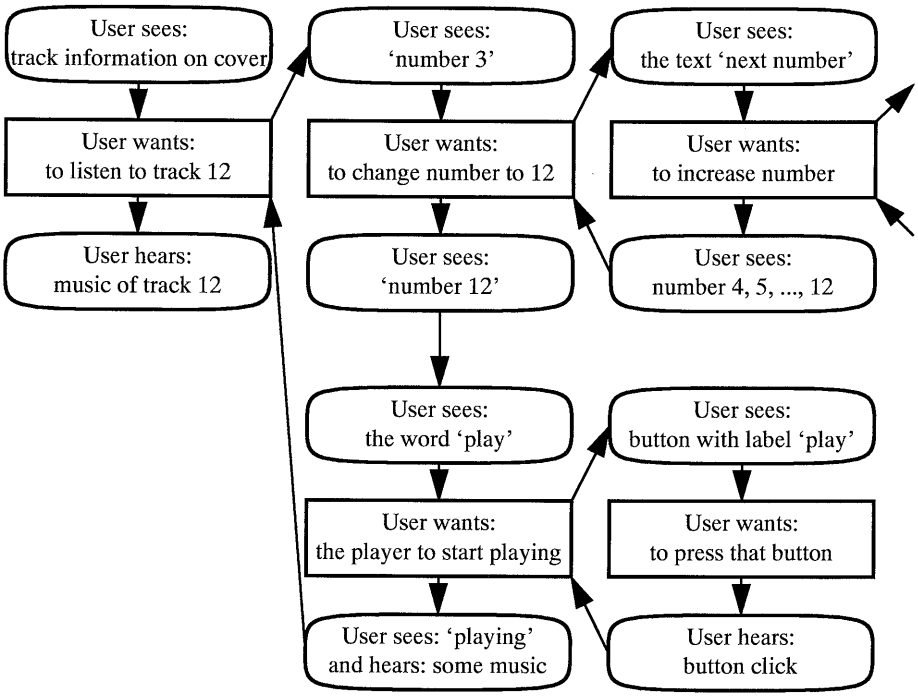


FIGURE 8. The intention hierarchy of the third user interface.

4.5 Three-step interaction

The fourth user interface also provides access to the recording functionality of CD-R players (see Figure 9). Users can play CDs as well as create new recordings. Unlike cassette recorders, CD-R players can only append new recordings after existing ones. Recordings cannot be erased. To simplify things, it is assumed that this CD-R player is part of an audio set comprising also a DCC (Digital Compact Cassette) player and a digital tuner. Since the set is fully digital, no recording level needs to be set. Users only have to indicate which device to record from. The CD-R player will set up the right connection and start that device.

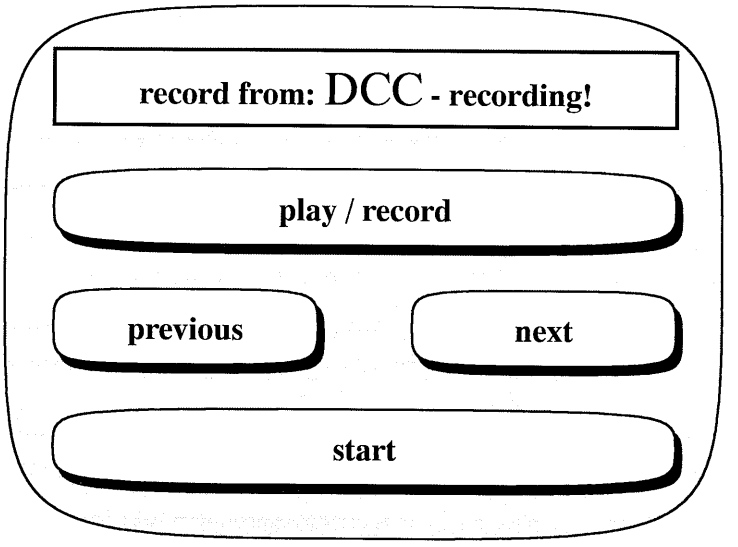


FIGURE 9. The fourth user interface also providing access to the recording functionality of the CD-R player.

First, users have to indicate whether they want to play from, or record onto the disc. The display indicates which function has been selected by reading 'play:' or 'record from:'. By pressing the 'play/record' button users can toggle between the two functions. When 'play' is selected a track number is indicated in the display; when 'record' is selected one of the available devices is shown ('DCC' or 'tuner'). Users can browse through the available tracks (in 'play' mode) or devices (in 'record' mode) by using the 'previous' and 'next' buttons. When the 'start' button is pressed the CD-R player will start playing or recording, showing 'playing!' or 'recording!' in the display instead of 'not started'.

From a PCT point of view, this user interface is interesting because the interaction now involves three qualitatively different steps: select play or record, change the track or the device and start the CD-R player. The interaction structure requires users to go through these three steps in the correct order.

Suppose a user wants to record from the DCC while the display shows 'play'. Now the only piece of information offered by the CD-R that is immediately associated with this intention is the label 'record'. Assuming that the positive association of the word 'record' outweighs the negative association of 'play', users are expected to press the 'play/record' button first because 'record' correlates best with their primary intention (the label-following heuristic).
After step one, users will see the word 'tuner' in the display, and realize that the wrong device has been selected. They are expected to form the intention of changing it to 'DCC'. Users will form this intention after the first step because only then will a device name be shown in the display. Showing a track number or a device name only when applicable prevents the risk of users forming irrelevant sub-intentions (information hiding). E.g. if the track number were shown permanently, users might inadvertently assume that the track number of a new recording needs to be provided, and worse, that new recordings can be positioned anywhere on the disc.
When the display shows 'DCC' users have fully specified what they want. Yet the CD-R player is not recording since it indicates 'not started' in the display. This should make users form the intention of starting the recording. The word 'start' prompts this. Because the word 'start' is not related to the overall goal, the action of pressing the 'start' button is expected to take place in the third and last step of the interaction (forward chaining).
All this should make users form the appropriate intentions in the correct order (see Figure 10).

This interface involves another issue: why would users use the 'next' or 'previous' button for changing 'tuner' to 'DCC'? Analogical reasoning (see e.g. Gentner, 1983) could provide an explanation. The assumption is that users have already learned to use those buttons for changing track numbers. Now analogical reasoning should make users consider using the *same buttons* because the word 'tuner' is at the *same position* in the display as the track number was previously.
Note that for changing the selected device the hill-climbing mechanism is not applicable because there is no indication of proximity. The sole reason for users to keep trying the 'next' and 'previous' buttons is that they change the correct part of the display. Until they reach the intended device there is no reduction in the distance to their goal. With a limited number of options this may work.

4.6 Discussion

Study
Four small user interfaces have been analyzed in detail. It was pointed out why novice users are assumed to effectively interact with the CD-R players right away. The motivation for this study came from user tests of two digital audio recorders (chapter 3: Eggen, Haakma & Westerink, 1996). In those tests it was observed that novice users, totally unfamiliar with digital recorders, could effectively use the play and tape-transport functionality right away. The question was what underlying mechanisms could make this happen? This chapter tries to answer this question by suggesting an explanation based on PCT for the interactive behavior of novice users when operating four simple user interfaces.

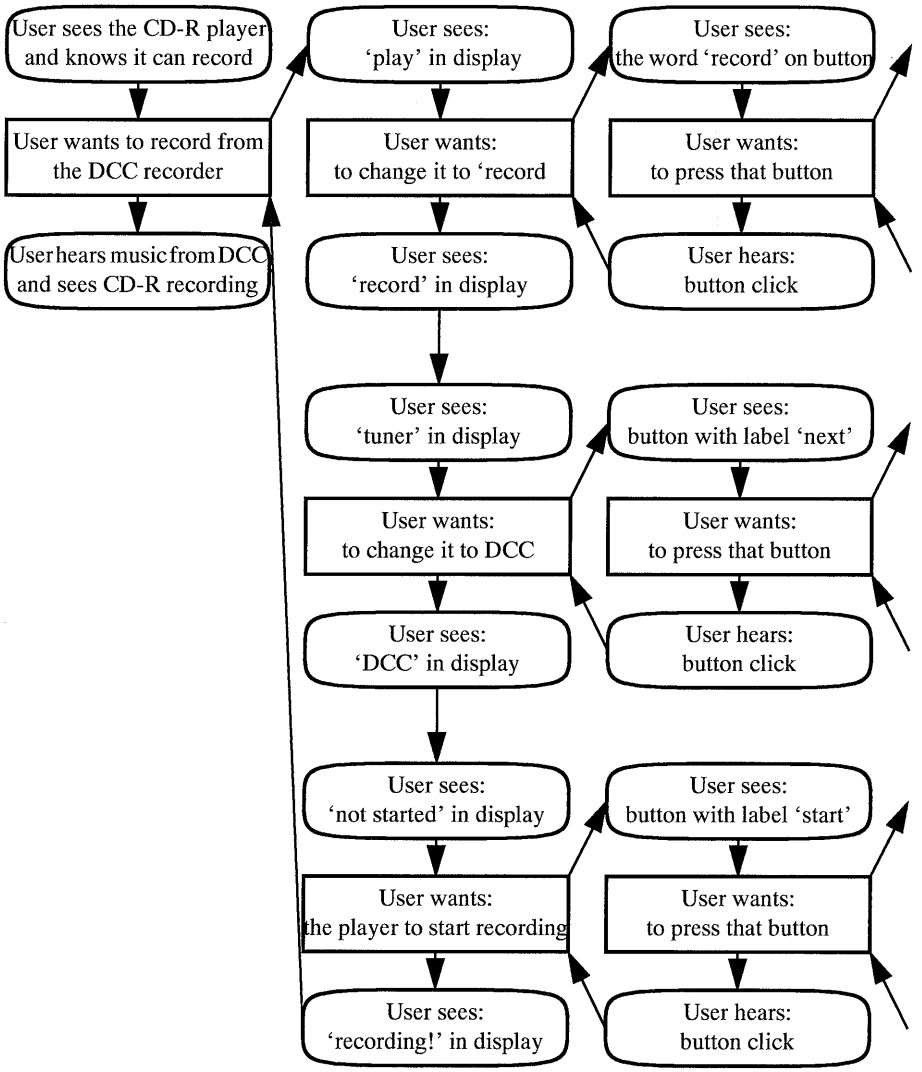


FIGURE 10. The intention hierarchy of the fourth user interface.

In this theoretical study it has not been tested whether users indeed easily learn how to use these interfaces. More importantly, it has not been verified whether the predicted user behavior corresponds to users' actual behavior. However, the four interfaces are based on the more complex digital audio recorders tested in chapter 3 (Eggen et al., 1996). Within that context, the interaction turned out to be easy to learn. The predicted user behavior was indeed observed in those user tests.

In predicting user behavior, assumptions are made about the effects of the E- and I-feedback on users. It is not hard to think of scenarios in which the effect of feedback upon users is totally different than predicted: all four interfaces are hard to learn for users who cannot read. Interfaces two, three and four are hard to learn for people who do not understand English. In order to envision the effects of E- and I-feedback on novice users, it is important to know about their background. The four interfaces are designed for users who can read English and are familiar with audio cassettes or CDs. They do not have to be familiar with digital audio recorders in general nor, of course, with these four interfaces in particular. The E- and I-feedback is expected to have the intended effects upon users with this profile.

The strength of this study is that it has proven possible to give a systematic and detailed explanation of the interactive behavior of novice users in the PCT framework. Some of the underlying psychological mechanisms have been identified which, in combination with the player's E- and I-feedback, are expected to trigger effective user behavior.

Related models

Perceptual Control Theory as used in this chapter has commonalities with other theories trying to explain user behavior, like Norman's four stages model (Norman, 1984), the CE+ model of Polson and Lewis (1990) and the Layered Protocols model (Taylor, 1988).

In the fundamental perceptual-control loop, shown in Figure 1, the different steps are named after the stages Norman distinguishes in user-system interaction. The mapping of the four stages onto PCT's control loop is straightforward. However, there are some conceptual differences. A first difference is in the definition of action selection and action execution. The *four-stages model* distinguishes between the two to separate users' mental activities from their physical activities. Within the PCT framework action selection is about the selection of relevant sub-intentions. This is linked to the intention-formation phase of the control loops one level down. The action-execution phase deals with all mental and physical activities aimed at achieving the selected sub-intentions. Within the context of PCT, the names 'sub-intention selection' and 'sub-intention execution' may be more appropriate than 'action selection' and 'action execution'.

Another difference between PCT and the stages model concerns the position of the evaluation stage. The stages model positions evaluation as the last user activity, emphasizing that users check the effects of prior actions. PCT positions this stage between intention formation and action selection, thus emphasizing that a difference between the desired and the actual perceptions provides the incentive for interaction. PCT views the user activity of checking effects of prior actions as a learning process that may cause longer-term adaptation of the user's behavior, but not as an element of the in-situ control loop. The PCT control loop is only about situated (inter-)action and does not comprise the element of retrospect.

A promising theory that explains how users learn from evaluating their actions against the observed system response is Lewis' *EXPL model* (Lewis, 1988). In a way, PCT and EXPL take a complementary approach. EXPL emphasizes how users explain a system response on the basis of the actions they have executed (retrospective) while PCT's E-feedback

aims at raising expectations in users about the effect of possible actions, thus assisting users in selecting effective actions (prospective).
EXPL has been incorporated as a learning component in the more general CE+ model (Polson and Lewis, 1990). Other components of the CE+ model are the problem-solving component and the execution component. Some of the mechanisms used in this chapter to explain the behavior of novice users, like hill climbing, forward chaining and label following, are also part of the problem-solving component of CE+. The execution component of CE+ indicates how users use their specific device knowledge and when users will fall back on problem-solving strategies. This part of the model resembles GOMS (Card et al., 1983) and CCT (Kieras and Polson, 1985).
By emphasizing the role of E-feedback for guiding user behavior, PCT contrasts with *GOMS* and *CCT*. Those models explain user behavior using production rules that users have internalized. Users have to learn those rules before being able to use the system effectively. Both types of models have their own application domain. PCT is more suitable for user-interface design of consumer applications to be used by novice and occasional users. GOMS and CCT seem to be more useful for professional applications that are used by skilled users on a regular basis.

PCT shares its hierarchical interaction structure with the *Layered Protocols* (LP) model (Taylor, 1992). PCT and LP stress that in order to let users form correct *intention* hierarchies, the *interaction* needs to be organized hierarchically: at each level of interaction information coming from the system allows users to form the correct intention hierarchy.
In chapter 2 (Engel and Haakma, 1993), a distinction was made between E- and I-feedback. I-feedback is information from the system to verify whether an intention has been satisfied. It reflects the system's *interpretation* of the user's actions. E-feedback is system information that raises *expectations* in users about the effect of possible actions. It assists users in selecting appropriate actions; see also Engel, Goossens and Haakma (1994) and chapter 3 (Eggen, Haakma and Westerink, 1996). The same types of feedback are introduced in the PCT control loop. In PCT terminology, the users' aim is to change perceptions created by I-feedback. Changing I-feedback is the purpose of user-system interaction. E-feedback, on the other hand, guides users towards effective interactive behavior. It assists users in forming correct intentions and provides guidance in achieving those intentions.

Analysis
For all four interfaces it has been analyzed how E- and I-feedback facilitate users to link intentions to sub-intentions and pursue sub-intentions in the correct order, which leads to effective user behavior. Four mechanisms have been identified for linking higher-level intentions to lower-level intentions: label-following heuristic (first interface), difference-reduction strategy (second interface), analogical reasoning (fourth interface) and information hiding (fourth interface).

- In the *label-following* heuristic the E-feedback shares some elements with the user's higher-level intention and hints at a lower-level action, e.g. the track numbers located on the buttons of the first interface.

- In the *difference-reduction* strategy the E-feedback raises expectations in users that the effect of the action will fulfil their intentions or will be a step in the direction towards fulfillment, e.g. the 'next' and 'previous' labels on the buttons of the second interface. The difference-reduction strategy requires that the E- and I-feedback at the higher level create a sense of distance or proximity in users.
- *Analogical reasoning* stimulates users to transfer procedural knowledge from one task to another. For example, in the fourth interface the same part of the display is controlled by the same buttons. However, the information displayed can be totally different: track numbers or audio devices.
- Information hiding prevents the risk of users forming sub-intentions that are irrelevant within the context of the overall task. The fourth interface does not show a track number when users have indicated they want to make a new recording and does not show an audio device when they want to play something. In this way the risk of users forming irrelevant sub-intentions is prevented and the likelihood of effective sub-intentions being pursued is consequently increased.

The difference-reduction strategy means that users will continue to try and reach their goals and not get discouraged when an action does not immediately lead to the effect ultimately intended. In the second user interface users are expected to keep pressing the next or previous button until they hear the intended track. The third interface illustrates that the difference-reduction strategy can work on several parts of an overall intention: changing the track number and giving the play command. In order to get users to pursue these sub-intentions in the correct order, the system has to indicate the dependency between the parts: E-feedback has to inform users, implicitly or explicitly, that the precise effect of pressing the play button depends on the track number in the display. In this way, users are enabled to link sub-intentions by forward or backward chaining. The fourth interface adds to this by requiring the formation of an extra sub-intention (pressing the start button) that is not immediately linked to the overall intention. This sub-intention will be pursued last because other sub-intentions seem more relevant to the overall goal.

4.7 Conclusions

Four simple user interfaces have been analyzed in the framework of Perceptual Control Theory. An account has been given of how the interaction is assumed to evolve according to PCT when novice users without any specific device knowledge start using those interfaces right away.

PCT advocates that the interaction between user and system be organized hierarchically. At every level the system should offer perceivable I-feedback about which users can form intentions. At every level E-feedback assists users in translating their intentions into sub-intentions one level down. The analyses indicate how the combination of E- and I-feedback at the various levels lets users build up correct intention hierarchies. They also show that label following (first interface), difference reduction (second interface) and analogical reasoning (fourth interface) are inclined to be among the mechanisms people use to

link intentions to sub-intentions. Confusion about relevant sub-intentions can be avoided when I-feedback is presented only when required (fourth interface). Some of the mechanisms for ordering sub-intentions are difference reduction (second interface) and forward and backward chaining (third and fourth interface).

Supported by indirect experimental evidence (chapter 3: Eggen et al., 1996), the analyses point out how Perceptual Control Theory can account for the interactive behavior of novice users.

4.8 References

Card, S.K., Moran, T.P. & Newell, A. (1983). *The psychology of human-computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates Inc.

Eggen, J.H., Haakma, R. & Westerink, J.H.D.M. (1996). Layered Protocols: hands-on experience. *International Journal of Human-Computer Studies,* **44**, 45-72.

Engel, F.L., Goossens, P. & Haakma, R. (1994). Improved efficiency through I- and E-feedback: a trackball with contextual force feedback. *International Journal of Man-Machine Studies,* **41**, 949-974.

Engel, F.L. & Haakma, R. (1993). Expectations and feedback in user-system communication. *International Journal of Man-Machine Studies,* **39**, 427-452.

Farrell, P.S.E., Hollands, J.G., Taylor, M.M. and Gamble, H.D. (forthcoming). Perceptual Control and Layered Protocols in Interface Design: I. Fundamental Concepts. *International Journal of Human-Computer Studies.*

Gentner, D. (1983). Structure Mapping: A Theoretical Framework for Analogy. *Cognitive Science,* **7**, 155-170.

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

Lewis, C.H. (1988). Why and How to Learn Why: Analysis-based Generalization of Procedures. *Cognitive Science,* **12**, 211-256.

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

Polson, P.G. & Lewis, C.H. (1990). Theory Based Design for Easily Learned Interfaces. *Human-Computer Interaction,* **5**, 191-220.

Powers, W.T. (1973). *Behavior: the control of perception.* Chicago, Illinois: Aldine Publishing Company.

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

Taylor, M.M. (1992). *Principles for Intelligent Human-Computer Interaction, a tutorial on Layered Protocol Theory.* North York, Ontario: DCIEM report no. 93-32.

Taylor, M.M., Farrell, P.S.E., Sempre, M.A.H. and Hollands, J.G. (forthcoming). Perceptual Control and Layered Protocols in Interface Design: II. The General Protocol Grammar. *International Journal of Human-Computer Studies.*

VanLehn, K. (1989). Problem Solving and Cognitive Skill Acquisition. In: Posner, M.I., *Foundations of Cognitive Science.* London: MIT Press.

Chapter 5

A user interface structured according to the Layered Protocols framework

Abstract

By providing a detailed description of a user interface for a Digital Compact Cassette recorder structured according to the Layered Protocols framework, this chapter illustrates how user-system interaction can be split up in a systematic way into a hierarchy of interaction protocols. It is indicated how the various interaction protocols relate and to what extent they can be designed independently of each other.
Structuring user interfaces in the way described in this chapter is advantageous in that it provides a systematic, recursive approach to user-interface design. Lower-level protocols can be designed and evaluated independently of the higher-level protocols, making it possible to detect, isolate and resolve usability problems early on. This is expected to save effort in comparison with the approach in which user interfaces are only designed and evaluated as a single unity.

5.1 Introduction

The Layered Protocols framework (Taylor, 1988 and 1992; chapter 2: Engel & Haakma, 1993; chapter 3: Eggen, Haakma & Westerink, 1996) structures user-system interaction in a hierarchical way. The interaction unfolds according to a hierarchically arranged collection of interaction protocols. Taylor (1988a) states that, for the most part, interfaces can be designed one level at a time. From a user-interface design perspective this is an attractive feature: user interfaces can be split up into components that are developed more or less independently.

By providing a systematic and detailed description of a user interface for a Digital Compact Cassette (DCC) recorder, this chapter explores how the interaction between user and recorder can be split up into separate interaction protocols and to what extent these interaction protocols can be designed independently of each other.

Users and interaction hardware
The DCC recorder is part of a digital hifi set. The interaction hardware comprises knobs and buttons for user input, and, for feedback, Light Emitting Diodes (LEDs) and a small dot-matrix display. The users are expected to be average people of about 20 to 40 years

old who understand the English language and have experience with analog cassette recorders. Thus, it can be assumed that users know what a cassette recorder is for, can read and understand English text and have no problems with pushing buttons and turning knobs.

Limitations
In order to limit the size of this chapter, only the interaction protocols will be described that make the recorder start accessing (e.g. playing) or changing (e.g. recording) the information on the tape loaded in the recorder. The protocols for loading cassettes will not be described, nor will the protocols detailing the interaction while the recorder is busy accessing or changing the information.

Structure
As indicated by the Layered Protocols (LP) framework, the user interface for the DCC recorder is structured as a hierarchy of interaction protocols; see Figure 1.
The top-level interaction protocol (*Tape*) allows users to manage the information stored on the DCC tape and offers the functionality for obtaining access to that information and changing it (e.g. for listening to the music and recording new music). The lower-level protocols (to the right in Figure 1) enable users to select one of the available functions and make the recorder execute it.
Some of those protocols are used for function selection (*CurrentTapePosition & CurrentTrack, Category, Play, Program, Rename, Edit, Source, TOC*[1] *and Name)*. These protocols allow users to observe and modify different pieces of information that, combined, select one of the top-level functions. To this end, these lower-level protocols build upon even lower-level interaction protocols of their own, which leads to the hierarchical interaction structure.
Some other protocols enable users to trigger function execution (**Push(Play)**, **Push(Execute)** and **Push(Confirm)**). In LP terminology, these protocols send a message to the top-level protocol. The top-level protocol interprets this message, taking into account the information handled by the other lower-level protocols, and responds by executing the selected function. The user can make these protocols send their messages by physically interacting with the recorder, e.g. by pressing buttons and turning knobs.

The interaction protocols shown in Figure 1 will be described in sections 5.2 to 5.9. The protocol descriptions will all feature the same components:

- *Information model*
  The information model describes, in an abstract way, what information is handled by the protocol.
- *Information presentation*
  This component indicates how the abstract information is presented to users.

1. TOC is the abbreviation of Table of Contents.

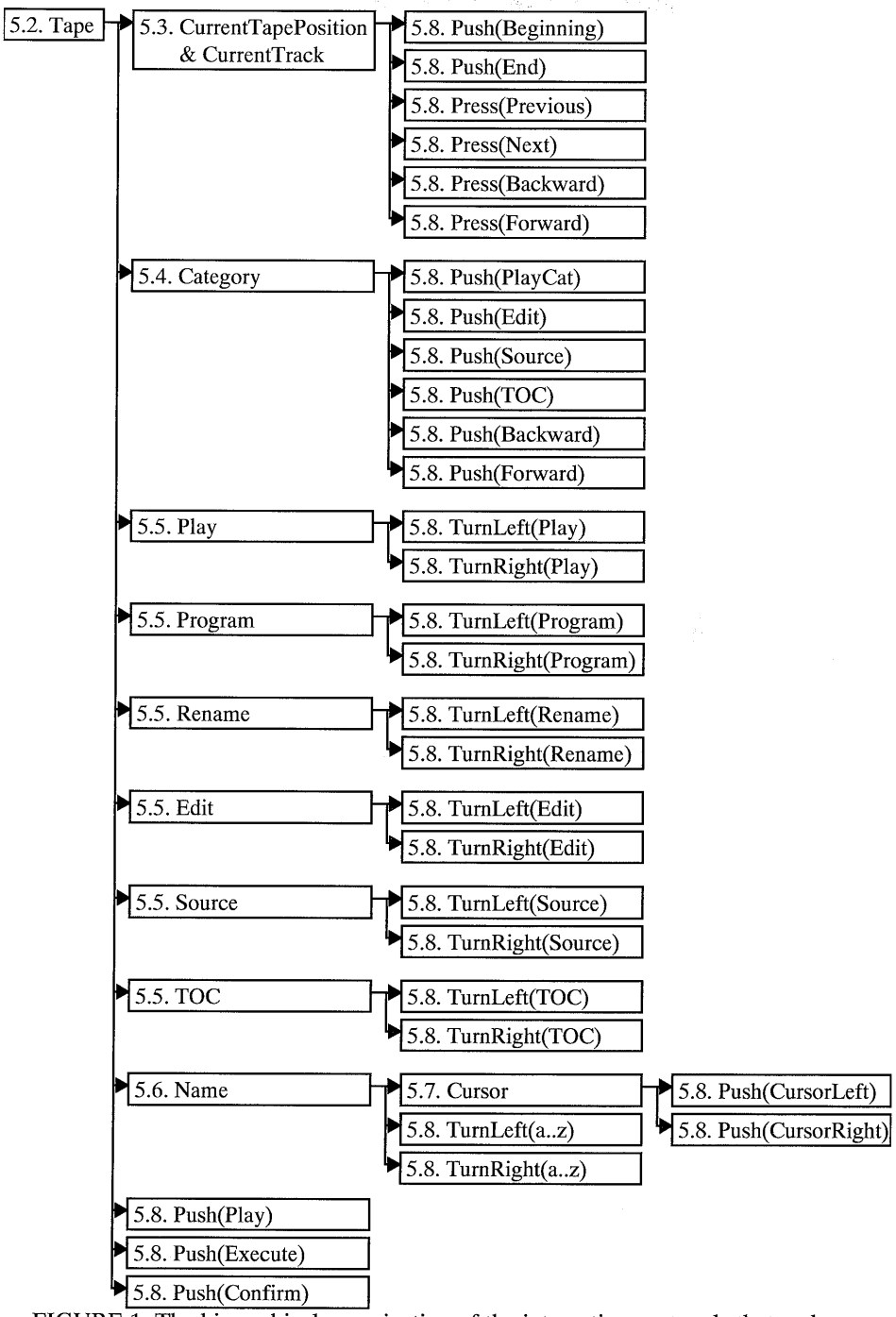


FIGURE 1. The hierarchical organization of the interaction protocols that make up the DCC recorder's user interface. The numbers indicate in what section of this chapter the protocol is described.

- *Functionality*
  This component lists the functions available to users for changing the information and the way it is presented.
- *Function selection and execution triggering*
  This part of the protocol description defines how these functions are selected and their execution is triggered. It is indicated what information is to be handled by lower-level protocols and what lower-level messages will trigger function execution.
- *Informing users*
  This component describes what information is offered to users to facilitate effective interaction (see chapters 2 and 4). It focuses on information for *intention formation* (information that helps users in determining what they want to achieve with respect to the information handled by the protocol), information for *action selection* (information that helps users in deciding how to achieve their intentions by indicating the relevant lower-level information and messages), and *action evaluation* (information that helps users in assessing the effects of the executed function).

The description of the user interface is top-down. The top-level protocol, dealing with the information on the tape, is described first. The bottom-level interaction protocols, dealing with the physical interaction, are described last.

5.2 Top level: the information on the tape

5.2.1 Information model

An inventory was made of the information that can be stored on a DCC tape. This is ultimately the information that is managed by users when interacting with the DCC recorder.

The following information is associated with a DCC tape:
- the length of the tape (e.g. 90 minutes);
- the name (a tape can be given a name like 'The Joy of Bernstein');
- and the information about the various tracks.

Each track is described by:
- a name (e.g. 'Gershwin: Rhapsody in blue');
- a starting position on the tape (e.g. 28'53" after the beginning of the tape);
- a length (e.g. 6'47");
- a number (e.g. the first track on the tape has track number 1); and,
- last but not least, the music (or, rather, the sound).

In addition, a music program is associated with a DCC tape. This program makes it possible for users to play a subset of the tracks on the tape. The music program indicates which tracks are in that subset and which are not.

For users, the music is the most important information on the tape: basically a cassette recorder is for recording and playing music. The other information helps users to find music to play or allocate space for new recordings.

5.2.2 Information presentation

Most of the information is visually presented to users; see Figure 2. The name of the tape is at the top, communicating which cassette has been loaded. Underneath is the name and number of one of the tracks. The length of the tape is indicated by the term 'D90'. It is graphically shown where the tracks are located on the tape, how long they are, and whether they are in the music program or not.

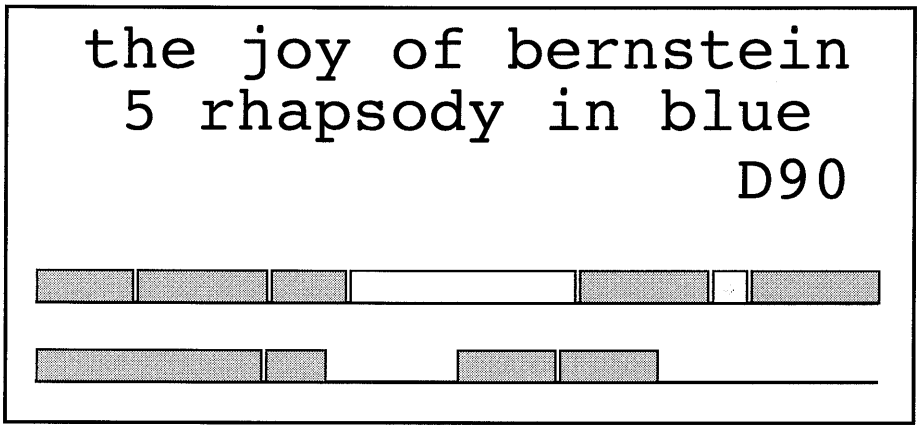


FIGURE 2. The visual presentation of most of the top-level information.
At the top is the tape's name. Underneath is the number and name of one of the tracks. The term 'D90' indicates that the tape is 90 minutes long. The tape is represented by two horizontal lines. The tracks are represented by boxes. The sizes of the boxes represent the lengths of the tracks. The tracks in the music program are displayed as grey boxes, while the others are white. The location of a track on the tape is indicated by the position of its box on the lines.

5.2.3 Functionality

Accessing tape information
Only the name and number of a single track is shown in the display. Since users may want to know the name of every track, an interaction mechanism is needed to determine what track number and name should be displayed. To this end, a variable is defined called *CurrentTrack*. This variable is handled by a lower-level interaction protocol; see section 5.3.

The music is presented auditorily. As with the track names, only a single piece of music can be presented at a certain moment. So, an interaction mechanism also has to be in place, scheduling the music to be played. The following function is available for this:

- **Play(From: <a tape position>, Tracks: <all** or **program only>)**
  This function plays the music located after the indicated tape position. If it should not play all tracks, it will only play the tracks in the music program.

Changing tape information
Besides having access to the information on the cassette, users may want to change that information. To accommodate this, the following functions are supported by the recorder:

- **Record(From: <a source>, Start at: <a tape position>)**
  This function makes it possible to record new tracks on the tape. One of the components of the hifi set must be selected as the source of the music. The tape position indicates where to start the recording. The record function stops when the end of the tape is reached or when the music ends.
- **Erase(From: <a tape position>)**
  This function erases (a part of) a track from the tape. The tape position must be at the beginning or inside a track. The erase function stops as soon as the end of the track is reached.
- **Split(At: <a tape position>)**
  This function splits a track into two adjacent tracks. The tape position indicates where the track must be split. It has to be inside a track.
- **Connect(At: <a tape position>)**
  This function makes a single track from two adjacent tracks. The tape position must indicate the end of the first track and the beginning of the second one.
- **Rename(What: <the tape or a track>, With: <a new name>)**
  This function changes the name of the tape or a track.
- **AddToProgram(Track: <a track>)**
  This function adds the indicated track to the music program.
- **RemoveFromProgram(Track: <a track>)**
  This function removes the indicated track from the music program.

It may happen that a cassette has no table of contents or that its table of contents is incorrect. This is unfortunate, because the user interface relies heavily on this information. To resolve this situation, the following functions are available:

- **CreateTOC**
  This function scans the entire tape to make an inventory of its information contents. After that it creates a new table of contents.
- **EraseTOC**
  This function deletes the current table of contents.

Frequency of use
Sooner or later, users may want to use each of these functions. However, some functions are expected to be used more frequently than others. Table 3 ranks the functions with respect to expected frequency of use.

Table 3: The top-level functions ranked with respect to expected frequency of use.

Ranking	Function
1	Play
2	AddToProgram, RemoveFromProgram
3	Record, Erase, Connect, Split, Rename
4	CreateTOC, EraseTOC

Impact
The effects of some functions can be easily undone. For example, the **Play** function does not have an effect on the tape information. Therefore, it is classified as a low-impact function. The **Record** and **Erase** functions, on the other hand, have a high impact because they may remove music from the tape that can only be recovered by re-recording it. Table 4 ranks the functions with respect to impact.

Table 4: The top-level functions ranked with respect to impact.

Ranking	Function
1	Record, Erase
2	Connect, Rename
3	Split, CreateTOC, EraseTOC, AddToProgram, RemoveFromProgram
4	Play

5.2.4 Function selection and execution triggering

An interaction mechanism has to be in place for users to select a function and trigger its execution. To this end, the lower-level interaction protocols and a decoding strategy have to be defined.
The design decisions taken here are largely arbitrary. There are several alternatives to the way in which the interaction has been structured here.

The objectives are that:

- users should be able to activate the **Play** function by pressing a single button. This is because the ***Play*** function has a high frequency-of-use in combination with a low impact;
- the activation of the functions that change some of the tape information takes at least two steps in order to prevent the risk of accidental execution. This is because these functions have a far lower frequency-of-use in combination with a higher impact.

Function selection
Nine different variables are defined for function selection, each handled by its own lower-level interaction protocol. Those lower-level variables are:

- *Category*
  This variable can have six different values: Play, Program, Rename, Edit, Record and TOC. The interaction protocol managing this variable will be described in section 5.4.
- *Play*
  This variable can have two different values: AllTracks and Program. Its interaction protocol will be described in section 5.5.
- *Rename*
  This variable can have two different values: Tape and Track. Its interaction protocol will be described in section 5.5.
- *Edit*
  This variable can have three different values: Erase, Connect and Split. Its interaction protocol will be described in section 5.5.
- *Source*
  This variable can have four different values: CD, Radio, Cassette and Other. This variable indicates the different sources that can be hooked up to the DCC recorder. Its interaction protocol is described will be section 5.5.
- *TOC*
  This variable can have two different values: Create and Erase. Its interaction protocol will be described in section 5.5.
- *Program*
  The *Program* variable can have two values: Add or Remove. Its interaction protocol will be described in section 5.5.
- *Name*
  Name indicates a new name to be assigned to the tape or a track. Its interaction protocol will be described in section 5.6.
- *CurrentTapePosition* and *CurrentTrack*
  CurrentTapePosition indicates a position on the tape. *CurrentTrack* indicates the track at the *CurrentTapePosition*. The interaction protocol managing the *CurrentTapePosition*, and thus the *CurrentTrack*, will be described in section 5.3.

Table 5 indicates how a function can be selected by setting these variables.

Table 5: Mapping between the lower-level variables and the selected function.

Lower-level variables	**Selected function**
Category=Play *Play*=AllTracks	**Play(From:** *CurrentTapePosition*, **Tracks:** all)
Category=Play *Play*=Program	**Play(From:** *CurrentTapePosition*, **Tracks:** program only)
Category=Program *Program*=Add	**AddToProgram(Track:** *CurrentTrack*)
Category=Program *Program*=Remove	**RemoveFromProgram(Track:** *CurrentTrack*)
Category=Record -	**Record(From:** *Source*, **Start at:** *CurrentTapePosition*)
Category=Edit *Edit*=Erase	**Erase(From:** *CurrentTapePosition*)
Category=Edit *Edit*=Split	**Split(At:** *CurrentTapePosition*)
Category=Edit *Edit*=Connect	**Connect(At:** *CurrentTapePosition*)
Category=Rename *Rename*=Tape	**Rename(What:** Tape, **With:** *Name*)
Category=Rename *Rename*=Track	**Rename(What:** *CurrentTrack*, **With:** *Name*)
Category=TOC *TOC*=Create	**CreateTOC**
Category=TOC *TOC*=Erase	**EraseTOC**

Execution triggering
Three lower-level messages lead to the generation of a top-level message triggering the execution of a top-level function: **Press(Play)**, **Press(Confirm)** and **Press(Execute)**. See section 5.8 for the lower-level protocols. The **Press(Execute)** message triggers the execution of the selected function right away. When **Press(Play)** is received, the *Category* variable is first set to Play (its default value). After that, the selected **Play** function is ex-

ecuted. This construct ensures that the execution of the **Play** function can be triggered in a single step, while triggering the other functions requires two interaction steps: selecting a *category* other than Play, followed by sending the **Press(Execute)** message.
The **Press(Confirm)** message provides an alternative way to trigger the execution of the **AddToProgram** and **RemoveFromProgram** functions.

Error conditions
In some situations, it may not be possible to execute the selected function. For example, the **Erase**, **Split**, and **Connect** functions cannot be executed when the *CurrentTape-Position* is at the end of the last track. In these situations, an error message is displayed indicating to users why a function cannot be executed and how this can be resolved.

The complete picture
Figure 6 shows the top-level interaction structure defined above.

5.2.5 Informing users

Now that the top-level design decisions have been made, it is time to consider how users are informed about this interaction protocol. Let us therefore take a look at what information is available to users for *intention formation*, *action selection* and *action evaluation*, and what effect this information is assumed to have upon users.

Intention formation
The information presented to users during function selection is assumed to stimulate intention formation in the following ways:

- The graphical tape representation makes users aware of the music that can be played and of potential locations for new recordings. Users start looking for the **Play** and **Record** functions because of their experience with analog cassette recorders.
- Users realize they can change the music program when they notice that the **Play** function has the option of playing a selected set of tracks.
- When only a single track name is displayed while the tape representation shows multiple tracks, users start looking for means to obtain access to the other track names.
- Users start looking for the **Rename** and **Edit** functions after they have made a recording and observe in the display that the result is not precisely what they had in mind. For example, tracks may not have received names, or the music may have been divided into tracks in a wrong way.
- The need for the **CreateTOC** function is assumed to arise when there is no tape representation in the display because the table of contents is not available. Users will have a need for the **EraseTOC** function when they see that the tape representation is inaccurate. It may be that users immediately want to create a new table of contents and skip erasing the old one.

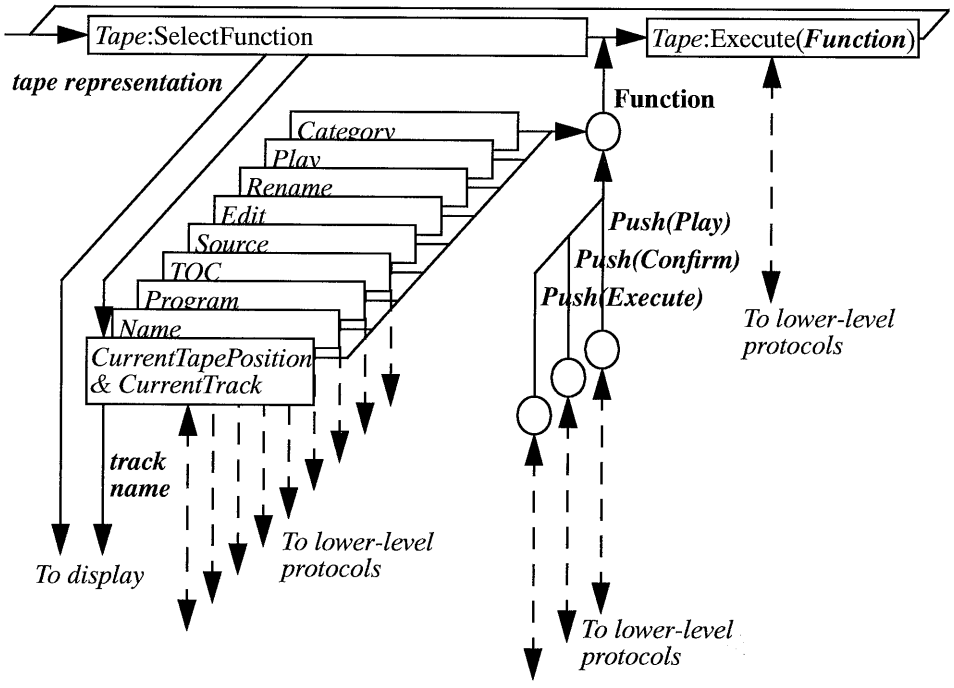


FIGURE 6. The top-level interaction protocol.
At the top level the information on the tape is handled: first, a user selects a function and triggers its execution. After that, the recorder executes the selected function. After function execution, users can start selecting a new function to be executed.
Nine variables are defined for function selection, each handled by its own lower-level interaction protocol. Function execution is triggered when one of the three Push messages is received from a lower-level protocol. When such a trigger message arrives, the function to be executed is determined and a message requesting the top level to execute that function is sent.
During function selection, a graphical tape representation is shown in the display. The lower-level variable *CurrentTrack* determines what track name is shown in the display.

Action selection
This section is to indicate the information offered to users for creating correct expectations with respect to the effect of their actions. Because this can hardly be done without reference to underlying interaction protocols, this discussion will be postponed until after all the sub-protocols have been defined; see section 5.9.6.

Action evaluation
After they have started the execution of some function, users should be able to check whether the recorder's response matches their expectations. So, the recorder has to com-

municate to users what it is doing.
The execution of most functions takes noticeable time. These functions require an interaction protocol of their own, which make sure users can track the progress in function execution and interrupt the execution when needed. In order to limit the size of this chapter, only an outline of these protocols will be provided.

A cassette recorder has only sequential access to the information on the tape. Therefore, the tape has to be transported to the correct position first. The current head position is shown in the tape representation by a grey arrow; see Figure 7. The target position is shown by a black arrow. Both positions are also indicated numerically. Users are made aware of the tape transportation process by the grey arrow moving towards the black arrow and the noise produced by the tape transport mechanism.

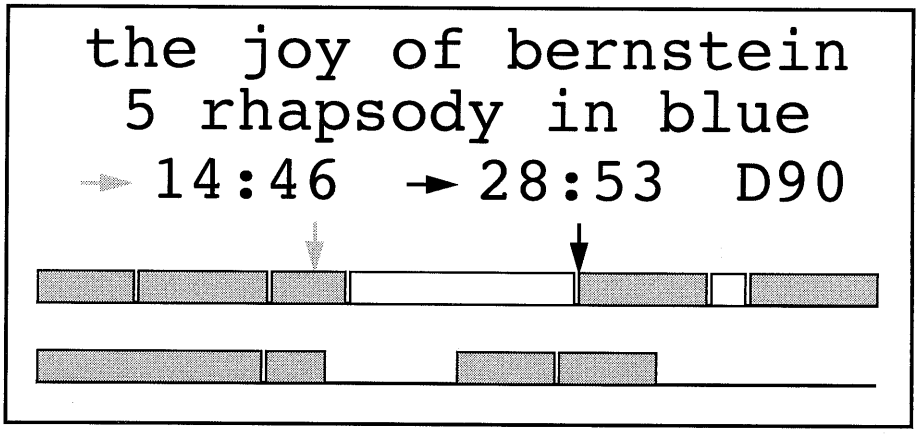


FIGURE 7. The recorder transports the tape to a target position.
The target position is represented by the black arrow. The grey arrow indicates the momentary head position. Their exact positions are also indicated numerically (28'53" and 14'46", relative to the beginning of the tape, respectively). The grey arrow gradually moves towards the black arrow. This creates a visual awareness in users of the tape transportation process. The transportation mechanism generates some noise, creating the same awareness via auditory clues. The noise stops when the grey arrow has reached the black arrow.

After the grey arrow has reached the black arrow, the actual function execution starts. The tape representation is constantly updated. This ensures that users can track how the recorder manipulates the tape. When the recorder is recording, users can observe the creation of new tracks. When it is erasing a track, that track gradually disappears. During the execution of **CreateTOC** the tape representation is gradually built up as the tape is scanned.
Auditory cues also play an important role. During **Play**, users hear the music on the tape. During **Record**, users hear the music being recorded. And during **Erase**, users hear a wip-

ing sound indicating that the tape is being cleared.
At any moment, users can stop function execution in the usual way, that is, by pressing the stop button.

Because function execution does not start right away, users may think that the recorder 'did not get the message'. To prevent this, a LED is switched on as an immediate response to a **Push(Play)**, **Push(Execute)** or **Push(Confirm)** message. This LED is switched off after function execution has finished.

5.3 CurrentTapePosition and CurrentTrack

5.3.1 Information model

The variables *CurrentTapePosition* and *CurrentTrack* indicate a position on the tape and the track at that tape position, if there is any, respectively. For example, the *CurrentTapePosition* may be 28 minutes and 53 seconds from the beginning of the tape, while *CurrentTrack*, the track at that position, is 'rhapsody in blue'.

Some tape positions are more likely to be selected by users than others. Table 8 ranks tape positions with respect to the likelihood of them being selected.

Table 8: Tape positions ranked with respect to frequency of use.

Ranking	Tape position
1	the beginning of the tape (for playing the entire tape)
2	the beginning of a track (for playing, renaming, erasing the track)
3	the end of the last track (for recording without erasing existing recordings)
4	the end of a track (for starting a recording after the last track to be preserved)
5	a position in the middle of a track (for splitting it)

5.3.2 Information presentation

The *CurrentTapePosition* is presented in the graphical tape representation as a black arrow.[1] Its value is also indicated numerically. The name of the *CurrentTrack* is shown in the display underneath the name of the tape; see Figure 7.

5.3.3 Functionality

On the basis of the tape-position ranking information, it was decided to provide six functions for manipulating the *CurrentTapePosition* and the *CurrentTrack*: **Beginning** and **End**, **Next** and **Previous**, and **Forward** and **Backward**.

- **Beginning** and **End**
  The function **Beginning** puts the *CurrentTapePosition* at the beginning of the tape. The function **End** puts the *CurrentTapePosition* at the end of the last track on the tape.
- **Next** and **Previous**
  The functions **Next** and **Previous** put the *CurrentTapePosition* at the beginning or end of a track. **Next** puts the *CurrentTapePosition* at the start or end position to the immediate right of the former *CurrentTapePosition*. **Previous** puts the *CurrentTapePosition* at the start or end position to the immediate left of the former *CurrentTapePosition*. After a small delay, this action is repeated. These functions keep repeating the action until the beginning of the tape or the end of the last track has been reached.
- **Forward** and **Backward**
  The functions **Forward** and **Backward** move the *CurrentTapePosition* forwards and backwards, respectively. They keep moving the *CurrentTapePosition* until the beginning or the end of the last track has been reached.

Frequency of use
Table 9 ranks these functions with respect to frequency of use.

Table 9: The functions ranked with respect to frequency of use.

Ranking	Function
1	Beginning
2	Next, Previous
3	End
4	Forward, Backward

Impact
All functions have the same impact. They all change the *CurrentTapePosition*.

1. The *CurrentTapePosition* indicated by the black arrow is only loosely coupled to the current head position indicated by the grey arrow in that the recorder will let the grey arrow follow the black arrow as fast as it can by transporting the tape. Traditionally, recorders do not distinguish between the two.

5.3.4 Function selection and execution triggering

Despite differences in frequency of use, it was decided to trigger the execution of the six functions in the same way, providing a uniform way for users to manipulate the *CurrentTapePosition*.

Function selection and execution triggering
The selection of a function and the triggering of its execution are conducted in a single step. No lower-level variables are introduced for function selection. Each function is associated with a lower-level message that triggers its execution; see Table 10. See section 5.8 for the lower-level protocols.

Table 10: The mapping of lower-level messages to the function executed.

Lower-level message	**Executed function**
Push(Begin)	Beginning
Push(End)	End
Press(Next)	Next
Press(Previous)	Previous
Press(Forward)	Forward
Press(Backward)	Backward

Error conditions
Beginning, **Previous** and **Backward** cannot be executed when the *CurrentTapePosition* is at the beginning of the tape. **End**, **Next** and **Forward** cannot be executed when the *CurrentTapePosition* is at the end of the last track. When users attempt to do this, an error sound will signal that the selected function has not been executed.

The complete picture
Figure 11 shows the interaction structure defined above.

5.3.5 Informing users

Intention formation
The black arrow indicates the currently selected tape position; see Figure 7. This arrow makes users consider that selecting a tape position may be a relevant sub-intention. Users will probably realize this because of their familiarity with analog cassette recorders. The graphical tape representation gives an overview of the available tape positions and is of help in identifying the tape position to be selected. Their experience with analog cassette recorders and compact-disc players should also make users look for 'forward', 'backward', 'next' and 'previous' buttons.

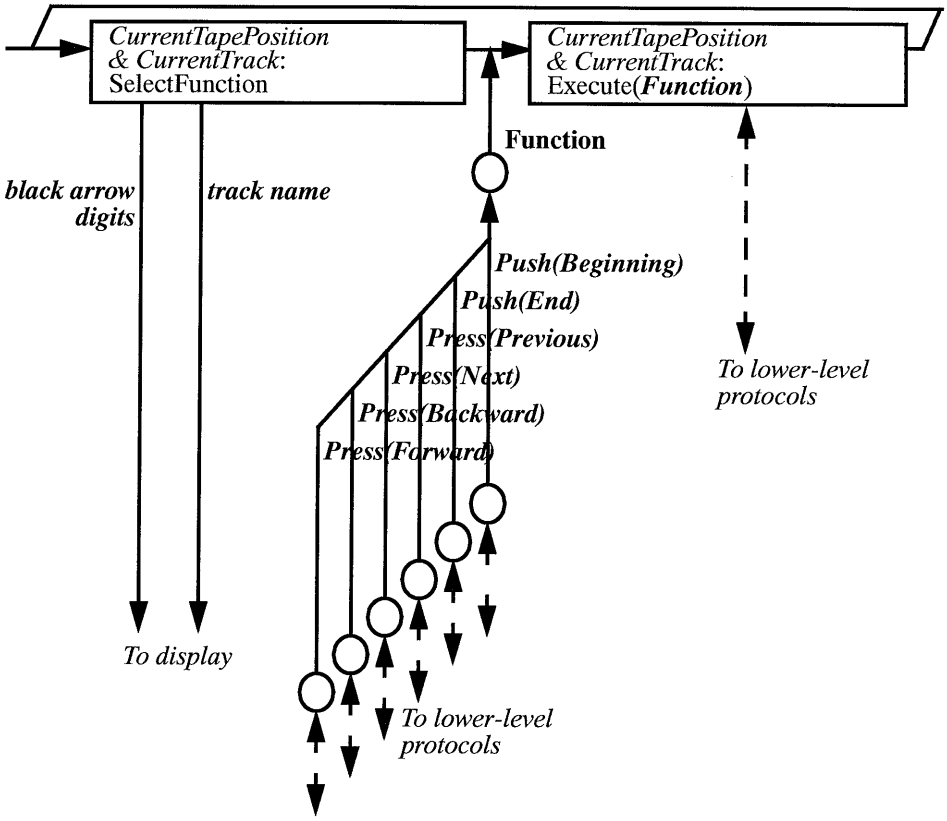


FIGURE 11. The interaction protocol for *CurrentTapePosition* and *CurrentTrack*. At this level the *CurrentTapePosition* is manipulated and, as a consequence, also the *CurrentTrack*. Function selection and execution triggering are integrated. No lower-level variables are introduced.
Six different messages are available for function selection and execution triggering, all generated by a lower-level protocol. When a message arrives, the function to be executed is determined and a message requesting the *CurrentTapePosition* level to execute that function is sent.
Users are informed about the *CurrentTapePosition* by a black arrow in the graphical tape representation. The *CurrentTapePosition* is also displayed numerically. The name of the *CurrentTrack* is shown in the display underneath the name of the tape.

Action selection
This discussion will be postponed until all the lower-level protocols have been defined; see section 5.9.5.

Action evaluation
The **Beginning** and **End** functions can be executed instantaneously. They do not require a separate interaction protocol because there is no opportunity for users to interact during function execution. The effects of the functions are observable to users. The black arrow has moved to the beginning of the tape or the end of the last track.

The other functions do require a separate interaction protocol because their execution takes observable time. By seeing the black arrow move, users observe progress in function execution. Users can interrupt the execution at any time by releasing the button they pressed for starting it.

5.4 Category

5.4.1 Information model

The *Category* variable can have any one of the values Play, Program, Rename, Edit, Record and TOC. The Play value is the default value, because the top-level **Play** function has the highest frequency of use.

5.4.2 Information presentation

Six LEDs are used to present the selected category to users. The LED associated with the selected category is on. The others are off; see Figure 12. From left to right, the LEDs correspond to the categories TOC, Record, Edit, Rename, Play and Program.

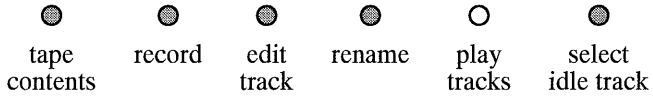


FIGURE 12. The LED associated with the selected category has been switched on, while the others are off.

5.4.3 Functionality

Six functions are made available to change the selected category: **SetCategory**(Play), **SetCategory**(Program), **SetCategory**(Rename), **SetCategory**(Edit), **SetCategory**(Record), and **SetCategory**(TOC). Those functions change the selected category to the corresponding value.

5.4.4 Function selection and execution triggering

Function selection and execution triggering
The selection of a function and the triggering of the execution are conducted in a single step. No lower-level variables are introduced. Each function is associated with a lower-level message that triggers its execution; see Table 13. See section 5.8 for the lower-level protocols.

Table 13: The mapping of lower-level messages to the executed functions.

Lower-level message	Executed function
Push(PlayCategory)	SetCategory(Play)
Push(Program)	SetCategory(Program)
Push(Rename)	SetCategory(Rename)
Push(Edit)	SetCategory(Edit)
Push(Record)	SetCategory(Record)
Push(TOC)	SetCategory(TOC)

Error conditions
There are no error conditions. The functions can always be executed.

The complete picture
Figure 14 shows the interaction structure defined above.

5.4.5 Informing users

Intention formation
One of the LEDs is on, the others are off. This makes users consider whether selecting a category is a relevant sub-intention. The top-level action-selection information should make this clear to users. Users have a good overview of the available categories because all six LEDs are visible at the same time. When users want to select another category, they are expected to look for means to switch on another LED.

Action selection
This discussion will be postponed until all the lower-level protocols have been defined; see section 5.9.4.

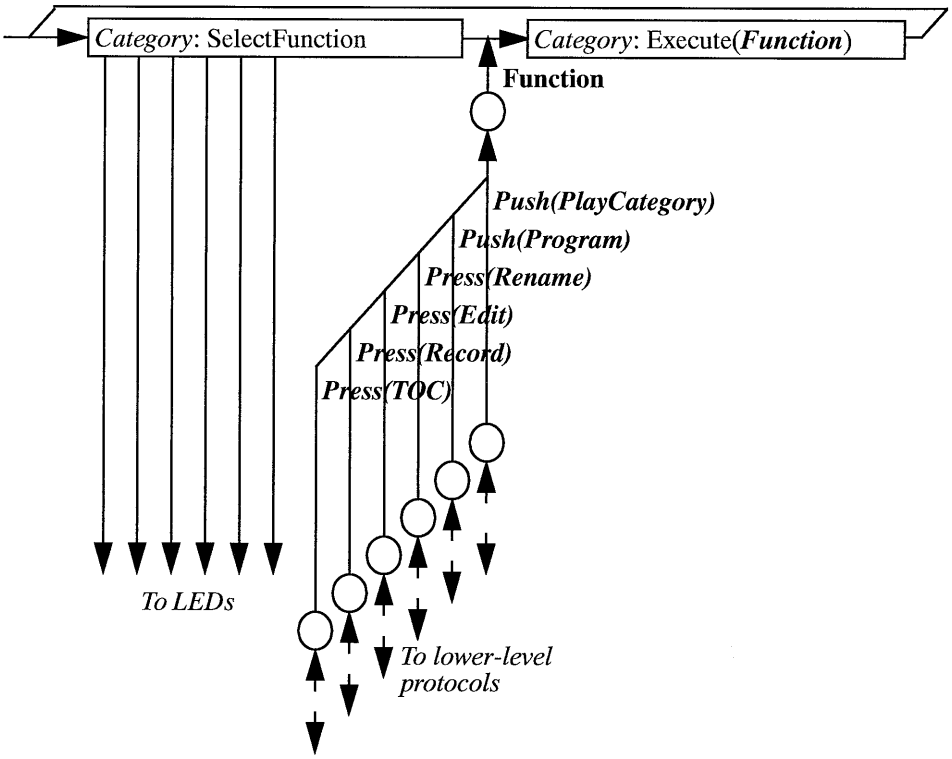


FIGURE 14. The interaction protocol for the *Category* variable.
At this level the *Category* variable is manipulated. Function selection and execution triggering are integrated. No lower-level variables are introduced.
Six messages are available for function selection and execution triggering, all generated by a lower-level protocol. When a message arrives, the function to be executed is determined and a message requesting the *Category* level to execute that function is sent. The LEDs show users the selected category.

Action evaluation
All the functions can be executed instantaneously. They do not require a separate interaction protocol because there is no opportunity for users to interact during function execution. Because one LED has been switched on and another switched off, the effects of the functions are immediately observable to users.

5.5 Play, Program, Rename, Edit, Source and TOC

Allthough their information contents differ, these six interaction protocols are structured in the same way. That is why they will be treated collectively in this section.

5.5.1 Information model

Table 15 shows the possible values for these variables.

Table 15: The possible values for the different variables.

Variable	**Values**
Play	AllTracks, Program
Program	Remove, Add
Rename	Tape, Track
Edit	Erase, Connect, Split
Source	CD, Radio, Cassette, Other
TOC	Create, Erase

5.5.2 Information presentation

An arrow is available for each variable to indicate the current value. Two of the arrows are shown in Figure 16.

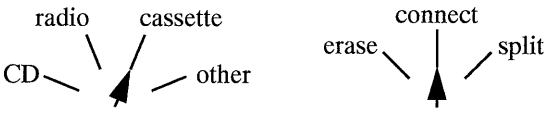


FIGURE 16. Two of the six arrows indicating the selected values.

5.5.3 Functionality

Two functions are available for each variable to change its value: **NextValue**(*variable*) and **PreviousValue**(*variable*). These functions change the value of the variable to its next and previous value, respectively. The values are ordered as indicated in Table 15.

5.5.4 Function selection and execution triggering

Function selection and execution triggering
Each function is associated with a lower-level message that triggers its execution; see Table 17. See section 5.8 for the lower-level protocols.

Table 17: The mapping of lower-level messages to the function executed.

Lower-level message	**Executed function**
TurnLeft(*variable*)	Previous(*variable*)
TurnRight(*variable*)	Next(*variable*)

Error conditions
When a variable has its first value, the **Previous** function cannot be executed. When a variable has its last value, the **Next** function cannot be executed. These error conditions never occur because the lower-level interaction protocols prohibit an erroneous message being generated when a first or last value has been selected.

The complete picture
Figure 14 shows the interaction structure of the *Source* variable defined above. The interaction structures of the other variables are the same.

5.5.5 Informing users

Intention formation
The variable's arrow points at one of the values. This makes users consider whether selecting a category is a relevant sub-intention. The top-level action-selection information should make this clear to users. Users have a good overview of the available values. They are shown on the front panel. When users want to select another category, they are expected to look for means to change the position of the arrow.

Action selection
This discussion will be postponed until all the lower-level protocols have been defined; see section 5.9.3.

Action evaluation
All the functions can be executed instantaneously. They do not require a separate interaction protocol because there is no opportunity for users to interact during function execution. Because the arrow positions change, the effects of the functions are immediately observable to users.

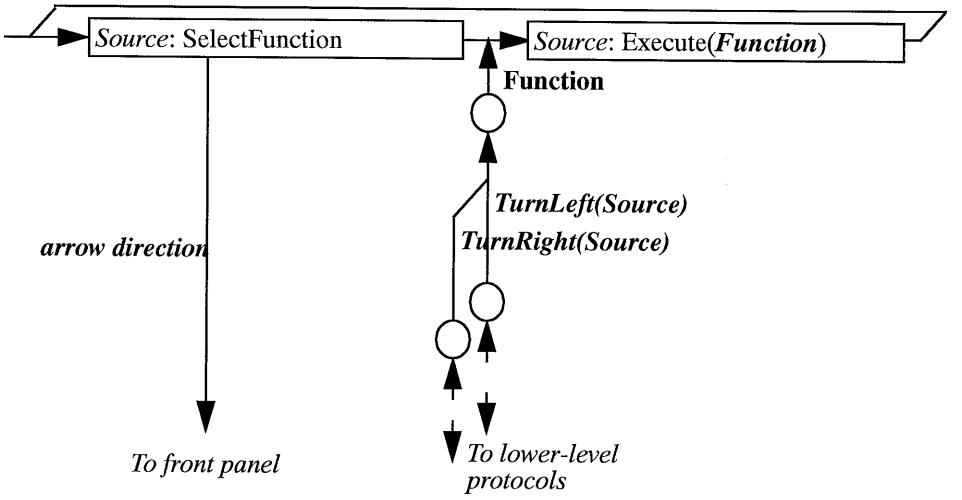


FIGURE 18. The interaction protocol for the *Source* variable.
This interaction protocol controls the *Source* variable. Function selection and execution triggering are integrated. No lower-level variables are introduced.
Two messages are available for function selection and execution triggering, both generated by a lower-level protocol. When a message arrives, the function to be executed is determined and a message requesting the *Source* level to execute that function is sent. The direction of the arrow indicates to users the selected value.

5.6 Name

The **Rename** function requires a new name as an argument. The variable *Name* supplies the new name. Its interaction protocol will be defined in this section.
The **Rename** function ranks low with respect to frequency of use. Therefore, the *Name* interaction protocol will be used only occasionally. Because of this, this interaction protocol is only activated when necessary, that is, when users have selected the Rename category (see section 5.4).

5.6.1 Information model

The *Name* variable is a text string. All the characters in the string are either a letter, a digit or a space.

5.6.2 Information presentation

The new track or tape name is shown in the display overlaying the old track or tape name.

5.6.3 Functionality

Usually text is entered using a keyboard. However, adding such an expensive piece of hardware to the DCC recorder solely for entering track and tape names is not justifiable. The interaction protocol defined here minimizes the number of buttons needed, at the expense of making entering text more long-winded.

Two functions are offered to change a character in the new name:

- **NextLetter(<the Nth character of the name>)**
  This function changes the Nth character of the name to the next letter in the character list {a..z, 0..9, <space>}.
- **PreviousLetter(<the Nth character of the name>)**
  This function changes the Nth character of the name to the previous letter in the character list {a..z, 0..9, <space>}.

Frequency of use
Both functions are rarely used. Once users start changing a character in the name, they are expected to continue so that for a while until they have found the character they want.

Impact
The **NextLetter** and **PreviousLetter** functions have the same impact. They reverse each other's effect.

5.6.4 Function selection and execution triggering

Function selection and execution triggering
A variable called *Cursor* is introduced to indicate which character should be changed. (For its interaction protocol, see section 5.7.) Two lower-level messages, **TurnLeft(a..z)** and **TurnRight(a..z)**, are introduced for triggering the execution of the **PreviousCharacter** and **NextCharacter** functions; see Table 19. The lower-level protocol for these two messages allows users to send them at a rapid pace, thus enabling users to go quickly through the character list; see section 5.8.

Table 19: The mapping of lower-level messages to the function executed.

Lower-level message	**Executed function**
TurnLeft(a..z)	PreviousCharacter(*Cursor*)
TurnRight(a..z)	NextCharacter(*Cursor*)

Error conditions
Potential error conditions arise when the beginning or end of the character list is reached. This is resolved by making it a circular list: ..., a..z, 0..9, <space>, a..z, ...

The complete picture
Figure 20 shows the interaction structure of the *Name* variable defined above.

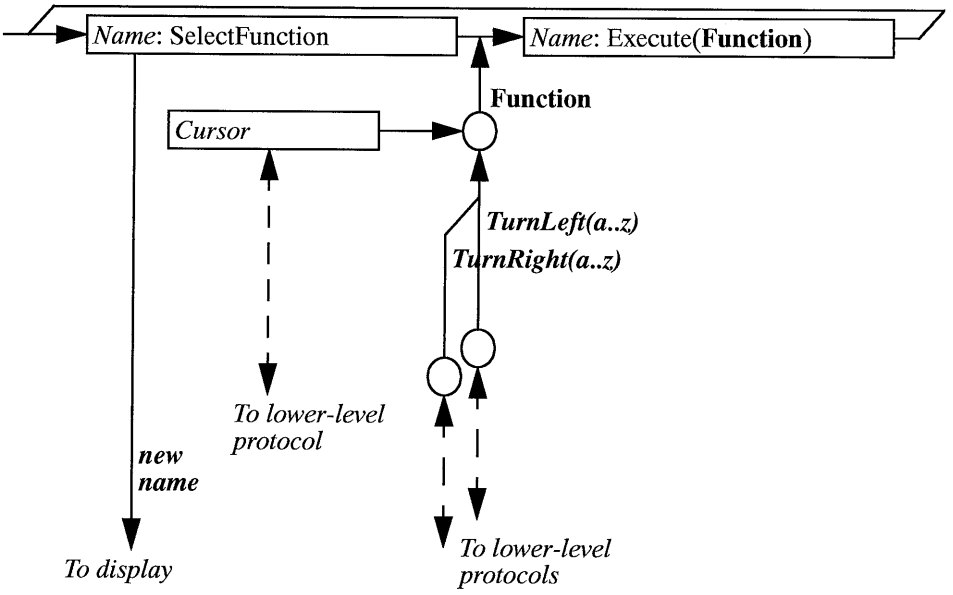


FIGURE 20. The interaction protocol for the *Name* variable.
This interaction protocol controls the *Name* variable. Function selection and execution triggering are integrated. One lower-level variable called *Cursor* is introduced.
Two messages are available for function selection and execution triggering, both generated by a lower-level protocol. When such a message arrives, the function to be executed is determined and a message requesting the *Name* level to execute that function is sent. The new name is shown in the display.

5.6.5 Informing users

Intention formation
The name of the tape and the name of the current track are shown in the display. When users see a name they don't like, they will want to change it.

Action selection
This discussion will be postponed until all the lower-level protocols have been defined; see section 5.9.2.

Action evaluation
All the functions can be executed instantaneously. The effects of the functions are immediately observable to users: a character in the name has been changed.

5.7 Cursor

5.7.1 Information model

The cursor indicates a character in the variable Name (e.g. the 4th character).

5.7.2 Information presentation

A horizontal bar underlines the currently selected character; see Figure 21.

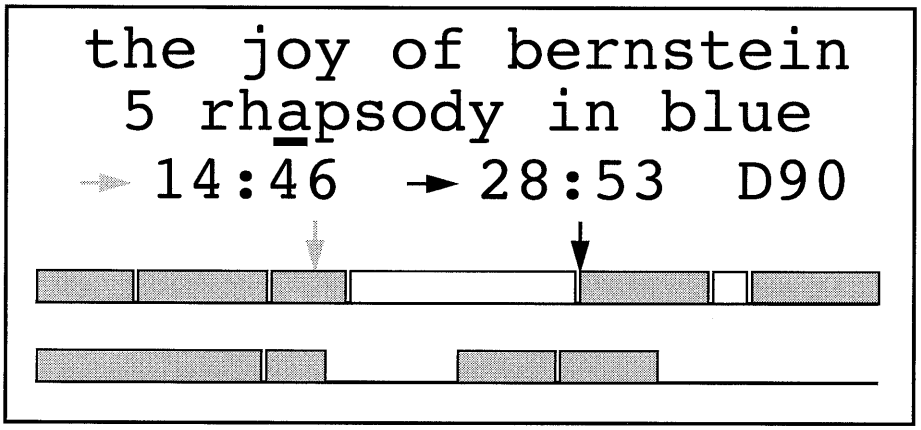


FIGURE 21. A horizontal bar underlines the currently selected character.

5.7.3 Functionality

Two functions are available to change the cursor position: **ShiftRight** and **ShiftLeft**. These functions shift the cursor's position one position to the right and left, respectively.

Frequency of use
The two functions have about the same frequency of use.

Impact
The **ShiftRight** and **ShiftLeft** functions have the same impact. They reverse each other's effect.

5.7.4 Function selection and execution triggering

Function selection and execution triggering
Two lower-level messages, **Press(CursorRight)** and **Press(CursorLeft)**, are introduced for triggering the execution of the **ShiftRight** and **ShiftLeft** functions; see Table 22. For the lower-level interaction protocols; see section 5.8.

Table 22: The mapping of lower-level messages to the function executed.

Lower-level message	**Executed function**
Push(CursorRight)	ShiftRight
Push(CursorLeft)	ShiftLeft

Error conditions
There is a maximum to the length of a name. The cursor cannot be shifted right when this maximum has been reached. And the cursor cannot be shifted left when the cursor is at the first character of the name. When users attempt to do this, an error sound will signal that the selected function has not been executed.

The complete picture
Figure 23 shows the interaction structure of the *Cursor* variable defined above.

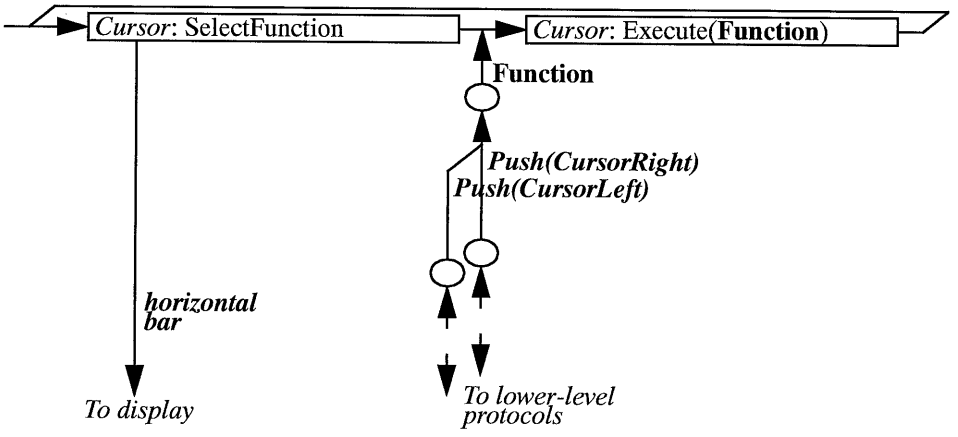


FIGURE 23. The interaction protocol for the *Cursor* variable.
This interaction protocol controls the *Cursor* variable. Function selection and execution triggering are integrated. No lower-level variables are introduced.
Two messages are available for function selection and execution triggering, all generated by a lower-level protocol. When such a message arrives, the function to be executed is determined and a message requesting the *Cursor* level to execute that function is sent. A horizontal bar in the display indicates the selected character.

5.7.5 Informing users

Intention formation
Users editing the *Name* will notice that the **TurnLeft(a..z)** and **TurnRight(a..z)** messages make the recorder change one particular character. This character differs from the other characters in the name in that it is underlined by a horizontal bar. This will make users want to move the horizontal bar underneath the character they want to change.

Action selection
This discussion has been postponed until all the lower-level protocols have been defined; see section 5.9.1.

Action evaluation
The two functions can be executed instantaneously. The effects of the functions are immediately observable to users: the position of the horizontal bar has changed.

5.8 Push, Press, Release, TurnLeft and TurnRight

This section describes the lowest level of interaction in which users immediately interact with the DCC recorder by pressing buttons and turning knobs. These interaction protocols differ from the other protocols in that they do not handle information. Instead, these interaction protocols generate messages that they send to the interaction protocols one level up, thus triggering the execution of a higher-level function.

5.8.1 Information model

Users interact with the recorder by pressing buttons and turning knobs.

5.8.2 Information presentation

All the knobs and buttons are visible on the front panel. Figure 24 shows what they look like.

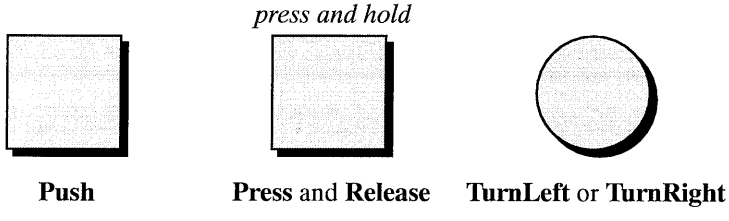


FIGURE 24. The three types of buttons and knobs that enable the physical interaction of the user with the recorder and the messages users can trigger with them.

5.8.3 Functionality

The left button allows users to generate the message **Push(<a button>)**. The middle button is used for generating the messages **Press(<a button>)** and **Release(<a button>)**. The knob to the right enables users to generate the messages **TurnLeft(<a knob>)** and **TurnRight(<a knob>)**.

5.8.4 Message selection and triggering

Users can trigger a message by carrying out the associated physical action:

- **Push(<a button>)**
  Users press the button and immediately release it.
- **Press(<a button>)**
  Users press the button and hold it down.
- **Release(<a button>)**
  Users release the button after holding it down.
- **TurnLeft(<a knob>)**
  Users turn the knob to the left.
- **TurnRight(<a knob>)**
  Users turn the knob to the right.

5.8.5 Informing users

Intention formation and action selection
Users see the knobs and buttons on the front panel. Users know how to use buttons because they are familiar with other electronic equipment. The label '*press and hold*' above the middle button explicitly invites users to hold the button down.

Action evaluation
Pressing and releasing buttons generates a clicking sound. So does turning knobs. This confirms to users that the message has been sent.

5.9 Action selection

Action-selection information raises expectation in users about the effect of an action. On the basis of this information, users can select an action that will bring them closer to meeting their intention. This section discusses the action-selection information made available to users at the different interaction levels. The information is located on the front panel of the DCC recorder, see Figure 30.

5.9.1 Cursor

The two buttons associated with the **ShiftLeft** and **ShiftRight** functions for manipulating the cursor position have labels indicating a left and right movement; see Figure 25. There is no explicit information indicating that these buttons cause the *cursor* to move.

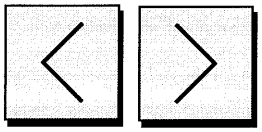


FIGURE 25. The labeling of the two buttons to be used for positioning the cursor.

5.9.2 Name

Users can trigger the execution of the ***NextCharacter*** and ***PreviousCharacter*** functions by turning a rotary knob to the left or to the right. The knob has no end positions. It can be rotated continuously. This ensures that users can rapidly scan through the list of characters.

The knob is labeled with the expression 'a..z, 0..9' indicating the available characters. There is no indication showing in which direction the button should be turned. It is also left to the user to note that the changing character is the one indicated by the cursor. The buttons for positioning the cursor are located close to the 'a..z, 0..9' knob. This should make it easier for users to make the link; see Figure 26.

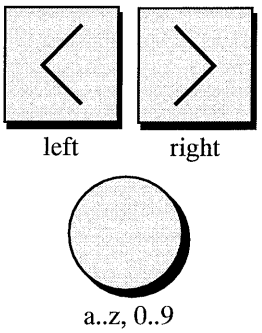


FIGURE 26. The buttons and the knob to be used for changing a name.

5.9.3 Play, Program, Rename, Edit, Source and TOC

The arrows are drawn on the knobs. This makes it obvious that turning a knob changes the position of its arrow; see Figure 27.

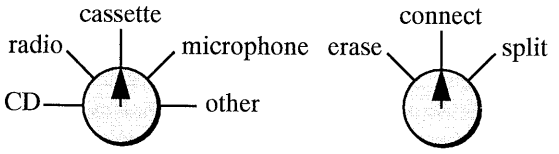


FIGURE 27. The arrows are located on the knobs.

5.9.4 Category

The LEDs used to indicate the selected category are integrated in the buttons for changing the selected category. This should make it obvious what button to use for selecting a specific category; see Figure 28.

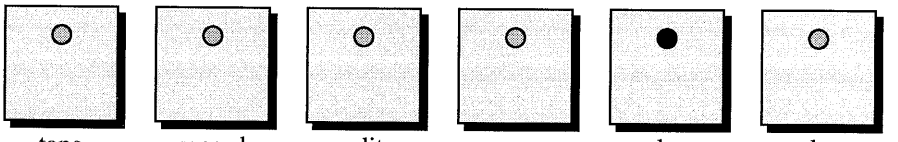


FIGURE 28. The LEDs used to indicate the selected category are integrated in the buttons for changing the selected category.

5.9.5 CurrentTapePosition and CurrentTrack

The six buttons and their labels used for changing the current tape position are shown in Figure 29. The labels and words are similar to the terminology and symbols used on CD players and analog cassette players for changing tracks and tape transportation. This should make users realize that those buttons can be used for changing the current tape position. The labels and words should lead users to form correct expectations about how the buttons affect the current tape position.

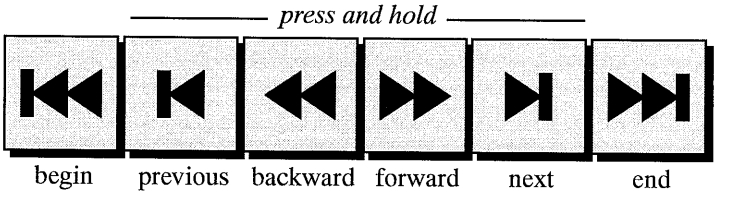


FIGURE 29. The labeling of the buttons for changing the current tape position.

5.9.6 Tape

The front panel of the DCC recorder is shown in Figure 30. It features the three buttons labeled 'execute', 'play' and 'confirm'. These buttons trigger the execution of a top-level function. Lines link the various buttons and knobs that are relevant for triggering the execution of a specific function. By combining the pieces of information along these lines, users should be able to form correct expectations about the effect of the three trigger buttons.

5.10 Discussion

The previous sections provided a detailed and systematic description of a user interface for a DCC recorder. The user interface is organized as a hierarchically arranged collection of interaction protocols. The different interaction protocols have been described more or less independently of each other. First, this section will discuss how interaction protocols relate to each other and to what extent they can be designed independently. After that, the benefits and risks in organizing user-system interaction in the way described in this chapter will be indicated.

5.10.1 Distinguishing and relating levels

Interaction protocols make it possible for users to access and change information. The information managed by the top-level protocol is what makes a system useful to users. The purpose of a cassette recorder, whether analog or digital, is to store and retrieve audio information. Sections 5.2.1 and 5.2.3 specifed the information and the related functionality that should satisfy the users' needs. These two paragraphs can be seen as the coupling point between user-requirements analysis, what facilities should the recorder offer users to satisfy their wishes, and user-interface design, how should users be provides access to those facilities. The information served as a starting point for the user-interface design describing the purpose of the top-level interaction protocol.

The information handled by the other, lower-level interaction protocols serves a purpose in the context of pursuing a top-level intention. For users, changing lower-level information is not a goal in itself. It is only useful in order to achieve a higher-level intention. For example, there is no reason for users to turn the *Source* button of the recorder unless they have the intention of making a new recording. The task analysis describes *what* users want to do, the user interface determines *how* they have to do it.
This creates freedom in the design of the user interface. Basically, users do not mind what lower-level information they have to work with, as long as it is made clear to them what role this information plays in managing the top-level information. To users, it is irrelevant whether they have to select the Record category and turn the *Source* button for recording music or whether they have to do it another way, as long as it is made clear to them that these actions are relevant for making a new recording. The user interface described in this chapter is one of a whole set of possible user interfaces for a DCC recorder. User interface

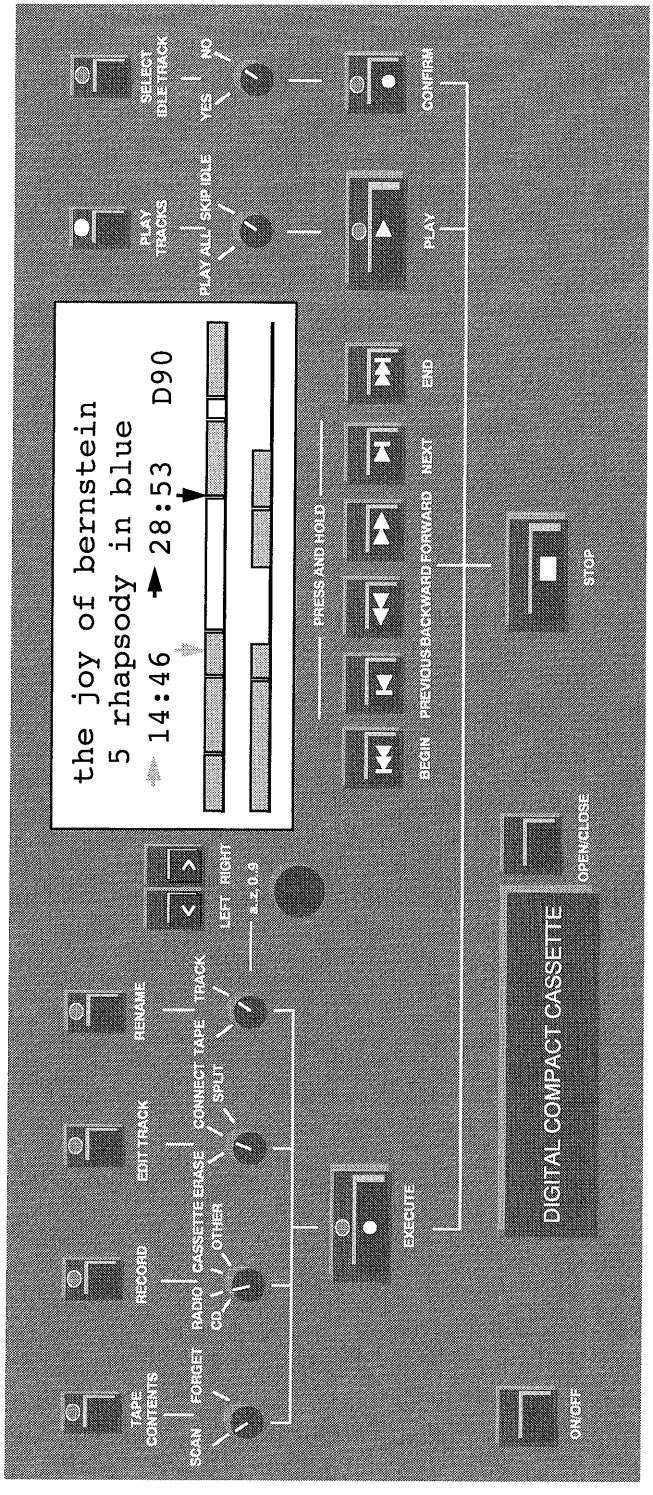


FIGURE 30. The front panel and display of the DCC recorder.

designers can organize the lower-level interaction protocols as they see fit, as long as it is made clear to users how these protocols support them in achieving their top-level intentions.

The LP framework points out that the gap between top-level information management and bottom-level physical interaction can be bridged in steps by introducing interaction protocols in between. An interaction protocol can build upon several lower-level protocols, prescribing *what* information should be handled by each of those sub-protocols and *what* messages they can send up. It is left up to the lower-level protocols to decide *how* to facilitate that. The LP framework treats the design of a lower-level protocol as a separate, smaller-scale user-interface problem, facilitating a systematic, recursive approach to bridging the gap.

The issues to consider are how to split up the interaction into a high-level protocol supported by a number of lower-level protocols and what the reasons can be driving those decisions. To come to grips with these issues, let us focus on the *Name* protocol that allows users to provide a new name for a track or the tape, and let us consider some design alternatives.
An impractical, but straightforward interaction design would be to have a button for each possible name. This design solution is impractical because of the huge number of buttons it requires. Another, equally impractical design solution is to offer users a single button to step through the list of possible names. This solution makes the interaction extremely long-winded, because it is a huge list. These solutions are feasible when choices are limited.
Both designs do not exploit that a name is structured as a string of characters. By introducing a cursor indicating a position in the string and the means to change a character, a name can be changed by changing its characters. An extra interaction level is created: putting the cursor at the correct position is made a relevant sub-intention. So, by introducing lower-level protocols that exploit the structure of the higher-level information, the number of steps in the interaction can be balanced against the amount of interaction hardware required.

Unfortunately, it is not always clear how to split up user-system interaction. For example, this user interface positions the *program* information at the top level. Another view is that this information serves as an argument for the **play** function and should therefore be dealt with at a lower level.
An argument for the first position is the life-time of the information. A music program is assumed to comprise the users' most valued recordings. This information is still valid after the music program has been played. It has the same life-time as the tape and should therefore be considered an integral part of the top-level information.
As far as the latter position is concerned, it can be pointed out that the music program information describes *how* users want to access the information on the tape and is not about *what* information should be on the tape. Therefore, the music program information should not be considered part of the top-level information, but instead be treated as a separate piece of information that serves as an argument for the play function to be handled by a separate lower-level protocol. This protocol could use the previous music program as the

default value for a new program, thus circumventing the life-time argument.
In this chapter, we took the first approach. Taking the latter position would have led to a different user-interface.

The user interface has to make clear to users what sub-protocols are introduced for what purpose. Action-selection information (see section 5.9) serves this purpose. It is information that helps users predict what the (higher-level) effect of an action (at the physical level) will be and what (lower-level) information influences that effect. The labels on the buttons hint at their effects; buttons effecting the same variable are grouped together; and lines connecting groups of buttons and knobs indicate what information is involved in determining the top-level function to be executed. This information makes it possible for users to select the actions whose (expected) effect is in line with their intentions. The action-selection information ensures that users can effectively bridge the gap between information management and physical interaction step by step. This is important because effectivity is the primary usability criterion (ISO, 1993).

A secondary, but nonetheless important usability criterion is efficiency: how much time does it take users to achieve their top-level intentions. Two of the factors influencing efficiency are how many physical actions are needed to trigger the execution of a function and how many physical actions are needed to reverse the effect of a function. A general guideline is that functions with a high frequency of use should require fewer physical actions than functions with a low frequency of use. This is why the **Play** function requires only a single button to be pressed.
Another guideline influencing efficiency is that accidental execution of functions that take many physical actions to reverse should be prevented. Though systems will never be able to tell a priori whether function execution is accidental or intentional, the risk of accidental execution of high-impact functions can be reduced by increasing the number of physical actions required to trigger that execution. This is why all functions changing the information on the tape require at least two buttons to be pressed. Someone randomly pressing buttons and turning knobs has a higher chance of triggering the **Play** function than triggering the **Record** function.
Besides the availability of interaction hardware, frequency of use and impact information drive decisions about what lower-level protocols to use for higher-level interaction.

5.10.2 Benefits

In this chapter it has been illustrated how a user interface can be described in a systematic way along the lines of the LP framework. There are several advantages to this approach.

- The user-interface design problem can be tackled in stages. Protocols can be designed more or less independently of each other. Since the role of an interaction protocol is clearly described, design alternatives can be considered in a more systematic way. The systematic approach makes it easier to see whether the user-interface specification is complete.

- Since sub-protocols are fully-fledged user interfaces in their own right, they can be tested by users in small-scale experiments. For example, a small experiment could point out whether users realize the difference between buttons that must be released immediately and buttons that must be held down, testing the effectiveness of the 'press and hold' label. Or, on a higher level, small experiments could test whether users will know how to position the black arrow (*CurrentTapePosition*) or provide a new name.
  These experiments can be done quickly. The functionality of sub-protocols is limited. Because of this, prototypes can be created relatively quickly and full-functionality user tests require relatively little effort in comparison with overall prototyping and testing. Top-level prototyping and testing will still be necessary, but only for identifying top-level usability issues. The lower-level usability problems can be detected and resolved early on.
- The effect of each piece of information upon users is described in the specification. When usability problems occur during testing, it should be straightforward to pinpoint the information that did not have the intended effect. Moreover, when a series of bottom-up tests are conducted, all usability problems can be attributed to a single interaction level. Interference between different usability problems is less likely to occur.

The risk in using the LP framework as applied in this chapter is that it is still unproven whether this interaction framework leads to user interfaces with good usability. A test conducted by Eggen, Haakma & Westerink (1996, chapter 3) indicated that this might well be the case. A similar benchmark test for this chapter's user interface could result in additional evidence.

5.11 Conclusion

By providing a detailed description of a user interface for a DCC recorder structured according to the Layered Protocols framework, this chapter has illustrated how user-system interaction can be split up in a systematic way into a hierarchy of interaction protocols. It has been demonstrated that the various interaction protocols can be designed more or less independently of each other. A higher-level protocol is linked to its lower-level protocols in that it defines *what* information is handled at the lower-level protocols and *what* messages they can send it. *How* users manage the information at the lower level and trigger the messages to be sent, is regarded as a smaller-scale user-interface problem that can be dealt with independently of the higher-level interaction protocol. It is up to the higher-level interaction protocol to decide *how* to interpret messages coming from lower-level protocols in the light of the information handled by these protocols and *how* to respond to them. It has been shown that design decisions about what lower-level protocols should support a higher-level protocol are influenced by the available interaction hardware, the expected frequency of use and the impact of the functions offered by the higher-level protocol. It is stressed that the user interface should communicate these decisions to users by means of action-selection information.

Structuring user interfaces in the way described in this chapter is advantageous in that it provides a systematic, recursive approach to user-interface design. Lower-level protocols can be designed and tested by users independently of the higher-level protocols. This is expected to involve less effort than the approach of designing and evaluating the user-interface as a single unity.

The author thanks René Wijnen and Gert-Jan Roskam for their help in designing the user interface and Frits Engel and Martin Taylor for providing valuable comments on earlier drafts of this chapter.

5.12 References

Eggen, J.H., Haakma, R. & Westerink, J.H.D.M. (1996). Layered Protocols: hands-on experience. *International Journal of Human-Computer Studies,* **44**, 45-72.

Engel, F.L. & Haakma, R. (1993). Expectations and feedback in user-system communication. *International Journal of Man-Machine Studies,* **39**, 427-452.

ISO (1993). Guidance on specifying and measuring usability, ISO CD 9241 part 11 (draft).

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

Taylor, M.M. (1988b). Layered protocols for computer-human dialogue II: Some practical issues. *International Journal on Man-Machine Studies,* **28**, 219-257.

Taylor, M.M. (1992). *Principles for Intelligent Human-Computer Interaction, a tutorial on Layered Protocol Theory*. North York, Ontario: DCIEM report no. 93-32.

Chapter 6

The usability of a user interface structured according to the LP framework

Abstract

An experiment is described that established the usability of a user interface for a Digital Compact Cassette (DCC) recorder. Its usability was compared with that of two other digital audio recorders with comparable functionality but different user interfaces. The DCC recorder was found to be more usable than the other two recorders, especially because its user interface took less time to learn.
The user interface was designed according to the Layered Protocols framework. Almost all of the considerations underlying its design worked out as expected. Those that did not could be explained within the framework. This confirmed the finding of chapter 3 (Eggen, Haakma & Westerink, 1996) that the Layered Protocols framework addresses important user-interface aspects impacting usability.

6.1 Introduction

In this chapter an experiment is described that assesses the usability of a user interface for a Digital Compact Cassette (DCC) recorder. A detailed account of the user interface in terms of the Layered Protocols framework (Taylor, 1988) can be found in the previous chapter. In the experiment, the usability of the user interface is benchmarked against that of two other digital audio recorders that offer comparable functionality but have different user interfaces. The usability of those two recorders had already been determined in the experiment described in chapter 3 (Eggen, Haakma & Westerink, 1996). The experiment described in this chapter established the usability of the DCC recorder in the same way as the usability of the other two recorders was assessed. The results are compared with the results of the first experiment.
In addition, it is discussed to what extent the different pieces of information provided to users by the user interface had the intended effect upon users described in the previous chapter.

6.2 Experiment

In the experiment described in chapter 3 (Eggen et al., 1996) two digital audio recorders were compared with one another with respect to their usability. The first recorder was a Digital Audio Tape (DAT) recorder. The other one was a prototype for a DCC recorder. (This DCC recorder will be referred to as 'the prototype' to distinguish it from the DCC recorder evaluated in this chapter.) For each recorder, 10 subjects executed a set of 8 tasks three times. For each task in each of the three sessions a task-completion score (indicating how well a task was executed) and a task-completion time (indicating how long it took the subjects to execute the task) were measured. In addition, it was recorded whether the manual was consulted. These measurements made it possible to compare the usability of the recorders in terms of effectiveness by comparing task-completion scores, in terms of efficiency by comparing task-completion times and in terms of learnability by comparing how the completion scores and execution times evolved over the sessions. In addition, a degree of subjective satisfaction was derived from the subjects' answers to a questionnaire. The usage logs and the video tapes of the experiments were used for more detailed studies. For a full description of the experimental details, see chapter 3.

To assess the usability of the newly designed DCC recorder, another group of 10 subjects was requested to execute three times that same set of 8 tasks: five relatively simple tape-transport and playing tasks and three more complex tasks (a programming, a recording and a tape-editing task). These subjects, 6 men and 4 women, had the same profiles as the subjects in the previous two groups. They were between 20 and 40 years old and had Dutch as their native language. Their educational background was non-technical and of a vocational or university level. They had no experience with digital audio recorders.
The subjects used a computer simulation of the DCC recorder. The simulation ran on a UNIX workstation and included auditory feedback and a front panel with real push-buttons positioned in front of the computer screen. The computer screen provided the visual feedback. The simulation was fully functional, except that there was no tray for loading and unloading cassettes. The simulation provided the subjects with a virtual tape. The 'open/close' button on the front panel functioned as a reset, leading to reloading of the virtual tape. The subjects had the impression of working with a real DCC recorder.

This third group of subjects performed the same experiment as the other two groups using a different digital audio recorder. Two aspects of the experiment could however not be replicated:

- one of the tasks, the programming task, requested to let the recorder automatically play a particular program over and over again. However, the DCC recorder did not offer such a repeat option. Instead, the subjects were requested to play the program starting at the beginning of the tape;
- in case of the DAT recorder and the prototype, a manual could be consulted on request. In case of the DCC recorder however, no manual was available. Instead of asking the experimenter for the manual, the subjects had the opportunity to ask for help.

The experiment was carried out using the same procedure. It consisted of three sessions. In each session, the same set of 8 tasks had to be completed. Before they started the first session, the subjects were given a written introduction of about 10 lines in which the most important aspects of the operation of the particular recorder being tested were indicated. They were allowed to study the introduction and explore the recorder for about 8 minutes. When they started the first session, the subjects were given the set of tasks and they were told that they could ask for help. The subjects were instructed to think aloud. If they had fruitlessly attempted to carry out a task for 20 minutes they were given the opportunity to proceed to the next task. At the end of the first session, there was a 20-minute break before the second session. At the end of the second session, the subjects were requested to fill in a questionnaire and they were explained the procedures for correctly executing the tasks. One week later, the third session took place.

The measurements were made the same way. A task-completion score was objectively assessed by the experimenter on the basis of explicit criteria and the time spent on each task was measured. Furthermore, it was recorded whether the subjects had received help. All the sessions were video-taped to enable closer investigation of the subjects' behavior and identification of the usability problems encountered.
These measurements made it possible to compare the interfaces in terms of effectiveness by comparing the task completion scores, in terms of efficiency by comparing the execution times and in terms of learnability by comparing how the completion scores and execution times evolved over the sessions. The degree of satisfaction was not directly assessed, but was derived from the answers to the questionnaire.

6.3 Results

The three recorders were compared on four aspects of usability: effectiveness, efficiency, satisfaction and learnability. The analyses were based on the new measurements for the DCC recorder in combination with the measurements obtained for the DAT recorder and the prototype (see chapter 3).

Effectiveness
An analysis of variance with repeated measures (MANOVA) was made of the task completion scores for the recorders, sessions and tasks. The main effects were found to be significant: recorder [$F(2,27)=6.8$, $p<.005$], session [$F(2,26)=19$, $p<.0001$] and task [$F(7,21)=9.4$, $p<.0001$]. Two of the two-way interactions were also found to be significant. The interactions between task and recorder [$F(14,42)=2.2$, $p<.05$] and between task and session [$F(13,15)=4.2$, $p<.005$] were significant. The other interactions were not significant at a 5% level.

The average task completion scores of the three recorders per session are presented in Figure 1. A more detailed analysis (comparing contrasts) of the data more precisely indicated the differences between the recorders and the sessions.
The task completion scores obtained for the DCC recorder were on average higher than those obtained for the DAT recorder. In the case of sessions 1 and 2 the differences were

significant [F(1,18)=4.4, p<.05 and F(1,18)=5.0, p<.05]. In the case of session 3 the difference was no longer significant at a 5% level. The task completion scores obtained for the DCC recorder are hardly any higher than those obtained for the prototype. In the case of all three sessions, these differences were not significant at a 5% level.
In the case of the DCC recorder, the increase in task completion scores from session 1 to session 2 was significant [F(1,36)=6.8, p<.05]. The difference between session 2 and 3 was not significant.

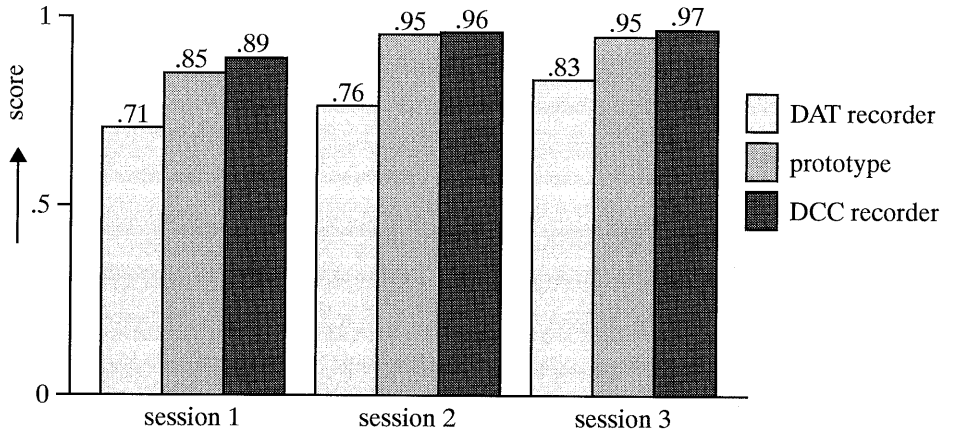


FIGURE 1. Effectiveness: the average task completion scores per session for the three recorders.

Efficiency
An analysis of variance with repeated measures (MANOVA) was made of the task completion times for the recorders, sessions and tasks. All the main effects were found to be significant: recorder [F(2,27)=27, p<.0001], session [F(2,26)=75, p<.0001] and task [F(7,21)=95, p<.0001]. The three two-way interactions were also found to be significant: between task and recorder [F(14,42)=5.8, p<.0001], between task and session [F(14,14)=4.3, p<.01] and between session and recorder [F(4,52)=4.1, p<.01]. Finally, the three-way interaction between task, recorder and session turned out to be significant [F(28,28)=1.9, p<.05].

The average amounts of time required by the subjects to complete the set of tasks are presented in Figure 2.
A more detailed analysis (comparing contrasts) of the task completion times revealed that for all three sessions the task completion times are significantly lower for the DCC recorder than for the DAT recorder [F(1,18)=22, p<.0001, F(1,18)=11, p<.005 and F(1,18)=22, p<.0001, respectively]. Only in case of the first session, the difference in task completion times between the DCC recorder and the prototype was significant [F(1,18)=11, p<.005]. In the case of session 2 and 3, these differences were not significant at a 5% level.

In the case of the DCC recorder, the difference in task completion time between session 1 and 2 was significant [$F(1,36)=12$, $p<.001$]. The difference between sessions 2 and 3 was not significant.

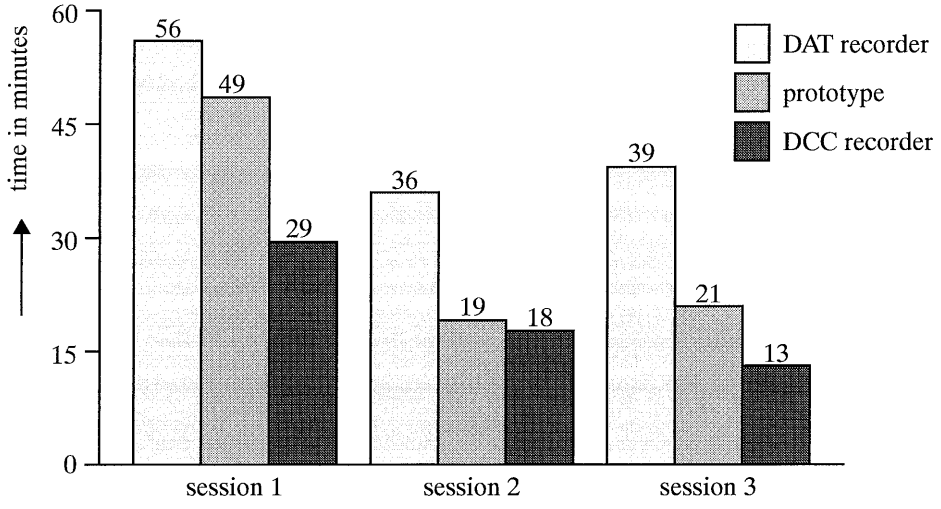


FIGURE 2. Efficiency: the average amount of time needed to complete the set of tasks per session for the three recorders.

Satisfaction

In general, the subjects who had used the DCC recorder expressed a more positive attitude in the questionnaire than the subjects who had used the prototype or the DAT recorder. This was most apparent when the subjects were asked what they thought of the experiment; see Figure 3. In the case of the DCC recorder, the nine subjects who answered this question unanimously said they liked the experiment. One subject did not answer the question. In the case of the prototype, only four subjects said they had liked the experiment, another four subjects indicated that they had eventually appreciated the experiment although they had experienced difficulties at first, while two subjects had found the experiment difficult to perform. In the case of the DAT recorder, two subjects said they had liked the experiment whereas eight subjects said that they had found the experiment difficult.

These results indicate a difference between the three recorders in terms of a subjective aspect of usability. The favorable results obtained for the DCC recorder were confirmed by the subjects' positive reactions after session 2 of the DCC experiment. At the beginning of the experiment the subjects had the impression that the DCC recorder was a complex device. After session 2 they generally felt very confident about the recorder. Some subjects even indicated that the experiment had given them sufficient confidence to start exploring their hifi-set at home.

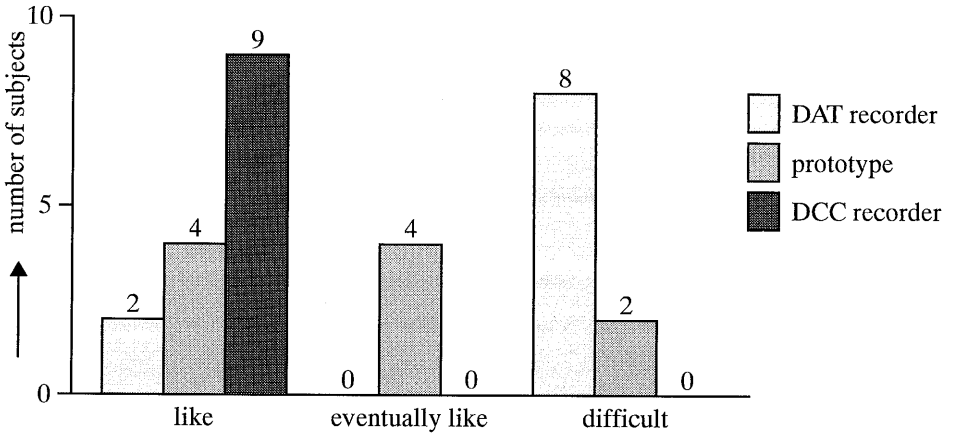


FIGURE 3. The number of subjects who liked the experiment, eventually liked the experiment or found the experiment difficult to perform for the three recorders.

Learnability

A learning effect was observed for all three recorders. Both the task completion scores and the task completion times showed a significant effect over the sessions. Overall, the scores increased over the sessions and the completion times decreased over the sessions. The issue now is to determine whether the recorders differ with respect to learnability.

The learnability of an interface is defined as the speed at which effectiveness increases over time. Figure 4 summarizes the data presented in Figures 1 and 2 and shows the average task completion scores as a function of the cumulative task completion times. This yields a comprehensive picture illustrating how quickly effectiveness increases over time. For each of the recorders it indicates how well the subjects on average mastered the recorder's operation as a function of the average amount of time they spent interacting with it.

Before the experiment, none of the subjects were familiar with the particular recorder tested or with digital audio recorders in general. In session 1, the scores obtained for the DCC recorder and the prototype were significantly higher than those obtained for the DAT recorder, while the completion times obtained for the DCC recorder were significantly lower than those obtained for the prototype and the DAT recorder. This means that, initially, the learnability of the prototype is better than that of the DAT recorder because subjects are more effective with the prototype after spending about the same time in interaction with the recorder, while the learnability of the DCC recorder is better than that of the prototype because subjects reach about the same level of effectiveness after spending less time in interaction with the recorder.

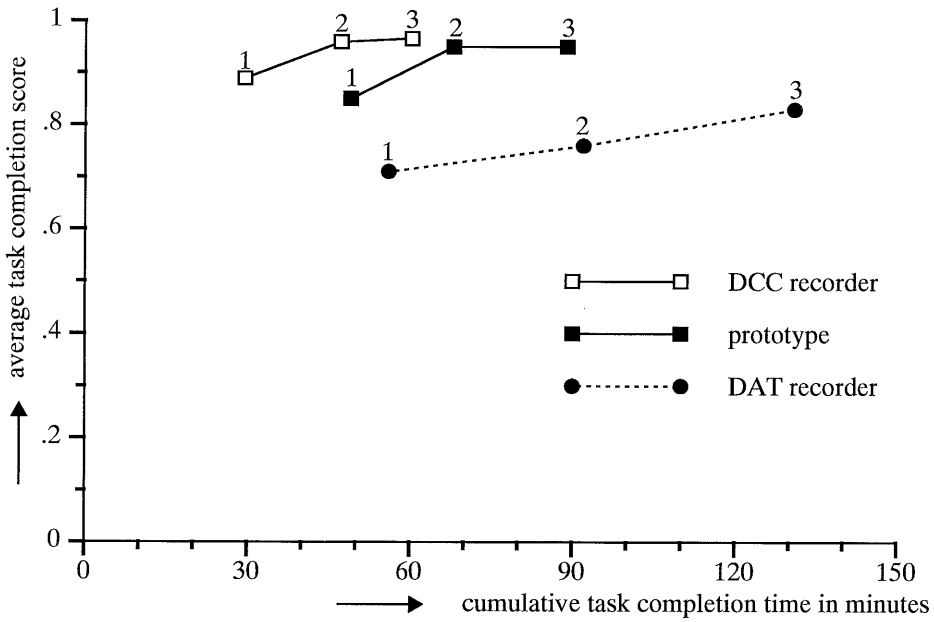


FIGURE 4. The 'learning curves' for the three recorders. For each session a data point has been plotted, representing the average task completion score for that session on the vertical axis, and the total time spent in interaction with the recorder (cumulative task completion times) on the horizontal axis. The graphs thus give an impression of how much was learned in what amount of time.

In session 2, the scores and the completion times obtained for the DCC recorder and the prototype were significantly better than those obtained for the DAT recorder. This means that the DCC recorder and the prototype are superior to the DAT recorder with respect to learnability.
In the sessions 2 and 3, the differences between the DCC recorder and the prototype in scores and completion times were not significant, even though the average scores and times obtained for the DCC recorder were consistently better. In these sessions, the subjects performed the tasks almost perfectly in about the same amount of time. This means that the difference in learnability between the DCC recorder and the prototype was entirely determined by the significant difference in task-completion times in the first session.

This difference in learnability was confirmed by the data regarding use of manual/request for help (see Figure 5). For the DAT recorder and the prototype it is indicated per session how often the subjects on average requested the manual. For the DCC recorder it is indicated per session how often the subjects on average asked for help. An analysis of variance with repeated measures (MANOVA) of these data revealed a significant difference between the three recorders [$F(2,27)=35$, $p<.0001$]. The subjects who were learning to use the DCC recorder needed hardly any additional help. The information provided by the user interface to a large extent sufficed.

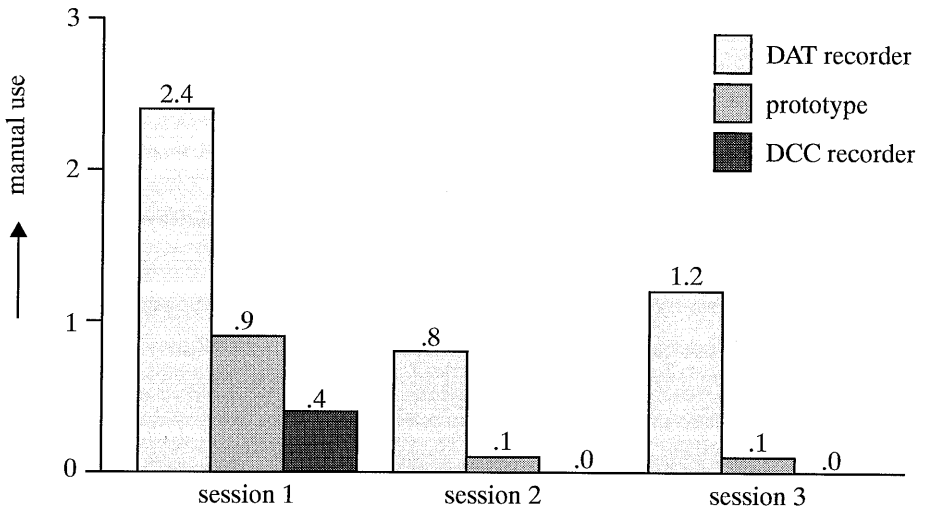


FIGURE 5. The average number of times per session that the subjects requested the manual (DAT recorder and prototype) or asked for help (DCC recorder).

6.4 Discussion

6.4.1 Attributing the found differences

The first issue to be discussed is whether the observed differences in usability between the DCC recorder and the other two recorders can be attributed to differences in the experimental set-up. As indicated before, the main differences in the experimental set-up were a minor change in the programming task and the lack of a manual.

In the programming task, the subtask of switching on the repeat option was changed for the subtask that the program was to be played from the beginning of the tape. This change was necessary because the DCC recorder did not offer the repeat option.
The influence of this change on the overall task-completion scores seems to be limited. The score of this subtask constitutes only 3% of the overall score. This means that the maximum influence of the change on the scores is limited to .03 points. And there is no reason to assume that the influence of this change is close to this maximum. The number of times the subjects successfully completed the 'repeat' or the 'beginning of tape' subtask was the same for the prototype and the DCC recorder: 26 out of the 30 times. (A comparison with the DAT recorder is not possible because in 14 out of the 30 times the subjects carried out the programming task they did not start executing the repeat subtask.) Along the same lines, the influence of the subtask change on the overall task-completion times also seems to be limited. Analysis of the video tapes proves that the execution times for the 'repeat' and 'beginning of tape' subtasks are short in comparison with those of the programming subtask. The programming subtask was the harder part. Generally, both the

'repeat' and the 'beginning of tape' subtasks caused no problems. The subjects were quick in recognizing the one or two actions required to execute them successfully.
For these reasons the influence of the subtask change on the task-completion scores and the task-completion times appears to be negligible.

The other difference in the experimental set-up was that, instead of requesting a manual, the subjects could ask for help. Help was verbally offered to subjects only four times. In these four cases, the help was offered at the experimenter's initiative, not at the request of a subject. (To remind subjects of this resource, the experimental procedure prescribed that the manual (or help) should be provided for the programming task in the first session if a subject had made no progress at all after about 10 minutes.) The subjects largely ignored the assistance. This is why the influence of the lack of a manual on the task-completion scores and the task-completion times seems not to have affected performance in any way.

Consequently, the differences found are most likely to have been caused by the different user interfaces. With respect to effectiveness and efficiency, the most usable recorder is the DCC recorder. The DCC recorder was found to be more efficient than the prototype in the first session, making the DCC recorder easier to learn than the prototype. The DAT recorder turned out to be less effective and less efficient than the DCC recorder. These findings are in line with the indications of satisfaction and the number of times additional assistance was required in the form of a manual or a request for help. Overall, the DCC recorder was found to be more usable than the prototype and much more usable than the DAT recorder.

6.4.2 Did the user interface work out as expected?

The user interface of the DCC recorder was systematically designed according to the Layered Protocols framework. In the previous chapter, the purpose of the different pieces of information provided to users by the user interface was identified. This section addresses the issue whether those pieces of information had the intended effect upon users.

Most of the time, the interaction between user and system evolved smoothly. This was already indicated by the experimental results and was confirmed by an analysis of the video tapes and the usage logs. From this we gather that by and large the subjects experienced no trouble in interpreting the state information provided to them, in forming intentions how to change that information and in selecting the appropriate actions to achieve those changes. The effects of their actions seemed to be in line with the subjects' expectations. Overall, the information offered to users for intention formation, action selection and action evaluation appears to have the intended effect, leading to a user interface that proves to be especially easy to learn.

However, there were also some critical incidents during the experiment, in which the interaction between user and recorder was not as fluent as expected. The three most crucial situations, which occurred at different levels of the interaction, will now be discussed.

Bottom level
The first issues occurred at the physical interaction level: only few subjects held down the 'press and hold' buttons long enough to discover their repeat feature. Most subjects immediately released these buttons and achieved a repeat effect by pressing them a number of times in succession. Exceptions were the 'backward' and 'forward' buttons. The subjects ended up holding those buttons down longer after they had observed that an immediate release had no effect.
This means that, in general, the 'press and hold' label accompanying this type of button did not have the intended effect upon users. This piece of action-selection information should be revised in order to better invoke the appropriate physical action.

Intermediate level
The selection of a top-level function involved the selection of one out of six function categories (Play, Program, Rename, Edit, Record and TOC). A Light Emitting Diode (LED) and a button were associated with each category. One of the LEDs was switched on, indicating the selected category. A category could be selected by using the associated button.
An unexpected observation was that subjects sometimes pressed the button associated with the category that had already been selected. From the video tapes it became clear that the subjects wanted to switch off the LED and de-select that function category. In particular, this behavior occurred when the Record LED was switched on and the subjects did not have to make a recording. In terms of the Layered Protocols framework, the subjects' intention to switch off the LED seemed to prevail over intentions relating to selecting another category.
The formation of such 'get-away' intentions had not been anticipated in the design. Pressing the button associated with the selected category had been designed to have no effect because users were expected to only form 'go-to' intentions relating to the selection of another category. The issue can easily be resolved. Pressing the button associated with the selected category should lead to the selection of the default (Play) category.

Top level
The third issue is related to the **CreateTOC** function. It may happen that a cassette has no table of contents (TOC) or that the information is incorrect. The **CreateTOC** function can be used to resolve this situation. It scans the cassette and creates a new table of contents.
The computer simulation of the DCC recorder's user interface was incomplete in that it did not provide visual feedback. The display was blank instead of showing the tape representation being gradually built up. Auditory feedback was provided, indicating that the tape was being transported at a high speed.
The omission of visual feedback was not expected to cause any problems. This function did not have to be executed during the experiment. However, three subjects initially thought that this function was relevant for the programming task, and were unable to determine the effect of this **CreateTOC** function. They never let the recorder finish the execution of this function.
This observation stresses the importance of feedback for action evaluation. According to

the LP framework, the subjects probably would have been able to determine the effect of the **CreateTOC** function and discard it for programming had they been provided with the appropriate visual feedback.

Had the user interface worked out as intended, the subjects would not even have considered using the **CreateTOC** function for programming. This indicates that the information for action selection did not have the desired effect. Table 6 provides more details on the misunderstanding between subjects and recorder. For each of the 10 subjects (A-J) it is shown how many times the subjects used the button or knob associated with the six function categories during their execution of the programming task of the first session.

Table 6: The number of times subjects (A-J) used a button or knob associated with the six different function categories during the programming task of session 1.

subjects **category**	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
Play	3	3	6	2	4	1	2	2	2	2
Program	19	5	2	2	7	2	2	1	1	5
Rename	-	-	-	-	-	-	-	-	-	1
Edit	12	-	5	-	-	-	-	-	6	7
Record	-	-	1	-	-	-	-	-	-	3
TOC	-	16	4	1	-	-	-	-	-	7

The programming task required the use of the Play and Program buttons and knobs. And indeed, all the subjects used those buttons. Probably due to the labeling, all the subjects considered the Play and Program hardware potentially useful for this task.
The subjects could not make any progress by using the hardware associated with the other four function categories. Five subjects (D through H) focussed on the appropriate buttons and knobs, while the five other subjects also used some of the other hardware. The latter especially considered the Edit and/or TOC hardware potentially relevant for the programming task. For example, subject A seemed to be really convinced that the Edit functions needed to be used. The idea was that a program could be defined by connecting the tracks to be programmed and erasing the remaining tracks. And subject B persistently tried to program using the TOC hardware assuming that the table of contents needed to be changed for programming.
Overall, these results are encouraging. Although they had never used a digital tape recorder before, all the subjects used the correct knobs and buttons the first time they encountered a more complex task. And half of the subjects used the correct ones only. However, the labeling of the Program, Edit and TOC buttons and knobs could probably be made more effective because half of the subjects considered the Edit and/or TOC hardware to be potentially relevant for the programming task.

6.5 Conclusions

The main conclusion of the experiment is that the DCC recorder is more usable than the prototype and much more usable than the DAT recorder. The main difference in usability between the DCC recorder and the prototype stems from a difference in learnability. In the first session, the subjects using the DCC recorder managed to do the tasks significantly faster than the subjects using the prototype while the tasks were executed about equally well.

Most of the considerations underlying the design of this specific user interface for a DCC recorder worked out as expected. The three most prominent points at which the interaction between user and recorder was not as fluent as expected could be explained in terms of intention-formation (bottom level), action-selection (top level) and action-evaluation (top level) information that did not have the intended effect upon the users. This indicates that the Layered Protocols theory addresses aspects of user interfaces that indeed influence usability.

No deficiencies in the LP framework have been found. The user interface of the DCC recorder showed no major usability defects.

The author thanks René Wijnen and Gert-Jan Roskam, for their help in designing, implementing and testing the DCC recorder, and Rudy van Hoe for his help with the data analysis.

6.6 References

Eggen, J.H., Haakma, R. & Westerink, J.H.D.M. (1996). Layered Protocols: hands-on experience. *International Journal of Human-Computer Studies*, **44**, 45-72.

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

Chapter 7

Recapitulation and conclusions

The central theme of this thesis is how the Layered Protocols (LP) framework can provide guidance in the design of full-scale easy-to-learn user-interfaces for novice and occasional users. This issue was divided into three research questions:

1. what are the central concepts of the LP framework and how do they relate to the usability of user interfaces, especially for novice and occasional users?
   The LP framework described by Taylor in his two 1988 papers is rather informal. Taylor introduced many potentially useful concepts, but it is unclear which concepts have most impact on usability. This question was addressed in chapters 2 and 4;
2. is it feasible to design full-scale user interfaces structured according to the LP framework?
   No full-scale application had been designed according to the LP framework in the consumer equipment domain. How to do this is a critical issue for the LP framework to have practical value. This question was addressed in the first half of chapter 3 and in chapter 5;
3. can it be experimentally verified that it is possible to design more usable interfaces with the LP framework and that the LP framework indicates aspects of user-system interaction that influence usability?
   If this is the case, the LP framework can be of help in developing more usable user-interfaces. The second half of chapter 3 and chapter 6 dealt with this issue.

In this chapter, the major findings of this thesis will be recapitulated and put into a broader context. After that, some conclusions will be drawn. Finally, an outline will be provided of how the LP framework could be developed further.

7.1 The LP framework and related UI models

The three most important aspects of the LP framework will be recapitulated and it will be indicated how they relate to concepts used in other user-interface models.

7.1.1 Layered interaction

The LP framework assumes a hierarchical task decomposition. In order to achieve their intentions, users execute sub-tasks which may consist of other sub-tasks. Other user-interaction theories also take this as a starting point, e.g. GOMS (Card, Moran and Newell, 1983) and Cognitive Complexity Theory (Kieras and Polson, 1985). The LP framework elaborates on this in that it organizes the *interaction* between user and system in a hierarchical way. It advocates that feedback should be provided at each level of interaction in

order to stimulate the development of mutual agreement about (intermediate) results. Interaction layers differ in the nature of the information exchanged between user and system.

The LP way of visualizing the hierarchical interaction organization is shown in Figure 1. In order to evoke a cooperative response from the system, users have to communicate their intentions. Intention transfer takes place along the thick arrows. Via successive stages, the main intention is encoded into a message at the lowest level. The message is relayed to the system via a physical transmission channel. The system in its turn successively decodes this message into the user's higher-level intention.
Feedback at each level (thin arrows) enables the user to verify the system's interpretations at all stages. The grey arrows indicate the exchange of virtual messages between the user and the system in successive layers. A virtual message is considered to pass on the contents of a series of real messages communicated by the underlying layers. For more details, the reader is referred to chapters 2 and 3.

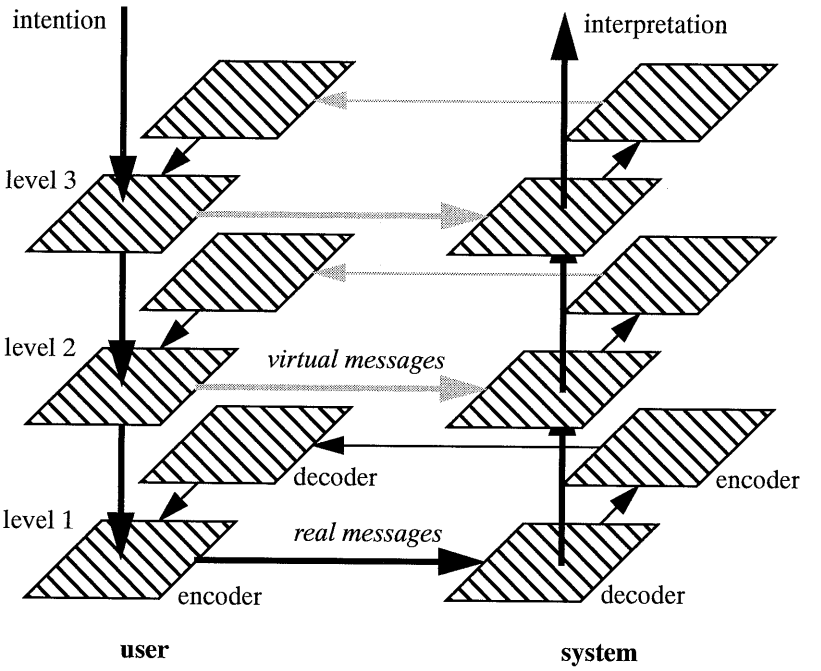


FIGURE 1. LP's hierarchical interaction structure.

7.1.2 Feedback at each layer: the Perceptual Control loop

To describe in more detail how the interaction between user and system evolves at each level, LP uses the Perceptual Control loop; see Figure 2 (Powers, 1973; Taylor, 1992). First users are expected to develop some idea about what they wish to perceive (intention formation stage). When not *actually* perceiving what they *want* to perceive, users may decide to interact with the system and try to realize the desired perceptions (evaluation stage). During the stage called sub-intention formation, users decide what they think the relevant sub-tasks are in order to reduce the discrepancy between what they currently perceive and what they want to perceive . In the last stage, users carry out these sub-tasks. This subintention-execution phase comprises all mental and physical activities aimed at achieving the selected sub-intentions. The users' physical activities provide input to the system (black arrow from user to system in Figure 2). To users, the system is a kind of black box hiding its internal operations. The black arrow pointing from system to user indicates the information generated by the system and perceived and evaluated by users (I-feedback). In addition, the user is assisted in intention formation and action selection by the provision of E-feedback (grey arrows). For more details, the reader is referred to chapter 4.

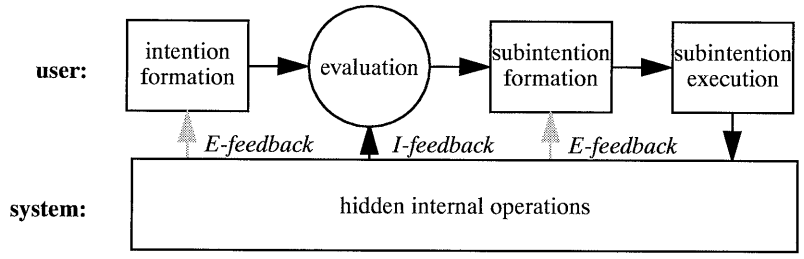


FIGURE 2. The perception-control loop with E- and I-feedback.

Interaction models, like Task Action Grammars (Payne and Green, 1986) and Cognitive Complexity Theory (Kieras and Polson, 1985), focus on modeling the interaction structure implemented by the system and on modeling the user's mental model of the task structure. These two interaction models indicate that effective interaction requires a good match between the two. However, little is said on how this can be achieved.
This is where the LP framework picks up. It assumes that mismatches are unavoidable. The user's model about how to use the system is likely to be incomplete and/or inaccurate. However, this does not necessarily lead to a breakdown of the interaction. By providing feedback at every level of interaction, the system's task model can be communicated to users in order to allow them to adapt their task model to the system's task model. In this respect, the LP framework approach is complementary to that of adaptive user interfaces which aim at adaptation of the system's interaction structure to the user's task structure.

The LP framework distinguishes between E- and I-feedback. I-feedback (the I stands for Interpretation) enables users to evaluate whether they perceive what they want to perceive. By providing I-feedback at every level of interaction, users are encouraged to form and evaluate a collection of potentially relevant intentions. E-feedback (the E stands for expectation) assists users in selecting from this collection of intentions the ones that are relevant for their overall task. By raising expectations in users about how some intentions are relevant for achieving others, users are encouraged to correctly link higher-level intentions to lower-level intentions. By providing E- and I-feedback at every interaction level, the system stimulates its users to form a hierarchy of intentions that links their overall goal to the relevant physical actions. Figure 3 symbolizes this.

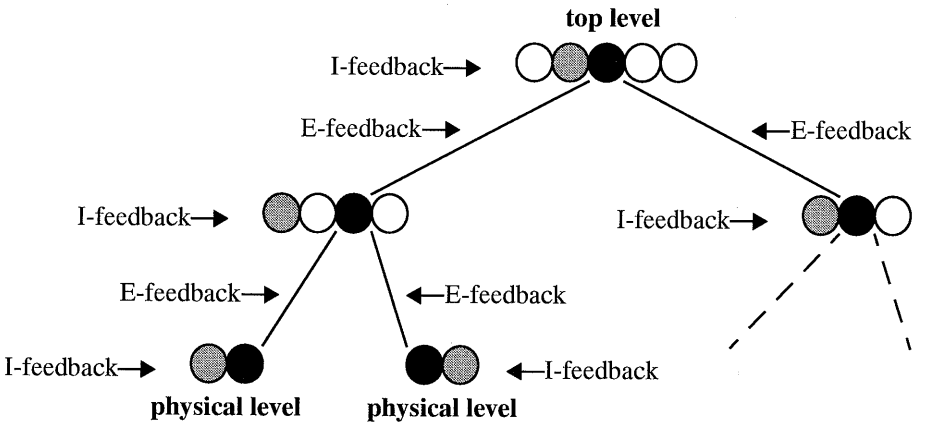


FIGURE 3. At every level of interaction, I-feedback allows users to evaluate whether the current situation (grey circles) matches their intention (black circles). E-feedback helps users in selecting those sub-intentions that are relevant for achieving their higher-level intentions.

7.1.3 Learning

The Perceptual Control loop bears resemblance to Norman's four-stage model (Norman, 1984). However, the stages model positions evaluation as the last user activity, emphasizing that users use system feedback to check the effects of their actions. By positioning evaluation as a finishing user activity, the stages model stresses the element of learning. By reviewing the effects of the latest actions, users learn the protocol rules underlying the user-system interaction. Task Action Grammars (Payne and Green, 1986) and Cognitive Complexity Theory (Kieras and Polson, 1985) take a similar approach. These two models define complexity measures that indicate how hard it is for users to learn to use the system on the basis of the number of interaction rules that users have to master. The problem with this approach is that immediate effective behavior of novice users, as observed in the second experiment (chapter 6), is hard to explain. According to these theories, users cannot be effective from the start because they first have to learn the effects of the available actions.

LP takes a different approach. Users should not have to learn what the effect of their actions *was*. Instead, it should be made clear to users what the effect of their actions *will be*. E-feedback prompts information that triggers expectations in users about the effects of potential actions. This information assists users in selecting effective actions and sub-intentions. By introducing E-feedback, LP emphasizes that user behavior, especially the behavior of novice users, is very much affected by the situations systems create during interaction.

This difference in approach can be nicely illustrated on the basis of Lewis' study that led to the EXPL model (Lewis, 1988), which was later integrated in a more general theory for easy-to-learn user interfaces called CE+ (Polson and Lewis, 1990). Lewis investigated how users generalize interaction procedures. He did this by first telling subjects what was initially on the screen, what actions were executed, and what the result of these actions was. With this information he would then ask subjects to explain the interaction rules. This line of questioning reflects the *retro*spective approach in which user behavior is assumed to be driven by (a generalization of) the interaction history. A similar study could be conducted in LP's *pro*spective approach. In such a study, subjects would not be informed about the action results. Instead, they would be asked what they think the results would be and why they expect that. The study would not be about generalization of learned interaction rules but about how specific expectations are raised in users.

To summarize, the LP advocates a hierarchical interaction structure for communicating to users the task structure implemented by the system. By providing I-feedback at every interaction level, users are stimulated to form intentions relating to what they ultimately want to perceive and the potentially useful sub-tasks. E-feedback aims at raising expectations in users about the effects of sub-task execution. In this way, E-feedback assists users in selecting the relevant sub-intentions that contribute towards the achievement of their intention at the next higher level. If E-feedback is provided at every interaction level, users should at every level be able to select the relevant sub-intentions and hence link their overall intention to the physical actions required to achieve it. Combined, the layered E- and I-feedback implicitly communicate the system's interaction structure to users. Because E-feedback aims at raising expectations in users, potentially reinforced by, but not necessarily acquired in, previous interactions with the system, the LP framework can be especially useful in the design of user-interface for novice and occasional users.

7.2 Applying the Layered Protocols framework

In addition to investigating the central LP concepts and how they relate to the usability of user interfaces, it was explored how to design full-scale user interfaces according to the framework. To this end, two user interfaces for DCC recorders were designed. The first user interface was hierarchically organized, providing E- and I-feedback to users at every level of interaction (see chapter 3). The design of the second user interface was more systematic in that each interaction protocol was specified according to a fixed format addressing the internal interaction structure as well as the way in which this structure was communicated to users (see chapter 5).

Chapter 5 illustrated that user-interfaces can be decomposed into a hierarchy of interaction protocols. The information controlled by the top-level interaction protocol provides users with a reason to interact with the system. This is the information that should make the system useful to users. The information controlled by lower-level interaction protocols is in itself not useful to users. It is only useful in that this information needs to be manipulated in order to enable manipulation of the top-level information. Lower-level interaction protocols were introduced because the number of functions (with all possible arguments) offered by the DCC recorder exceeded the number of knobs and buttons available to select these functions and trigger their execution. A one-to-one mapping between functions and buttons was not possible: pressing the same button had to result in the execution of different functions depending on some contextual information. It was the information controlled by the lower-level interaction protocols that supplied this contextual information.

In chapter 5 it was indicated that design decisions relating to what lower-level protocols are introduced to support a higher-level protocol should take the frequency of use and the impact of the higher-level functionality into account. When higher-level functions are expected to be used frequently, the number of interaction steps should be minimized in order to make the interaction as efficient as possible for users. When higher-level functions have an effect that users cannot easily undo, the risk of accidental execution of these functions should be minimized. This can be achieved for example by introducing an extra interaction step in which the system checks whether the user really wants the selected function to be executed. When a function combines a high impact with a high frequency of use, it is advisable to reduce the impact of the function by introducing an additional function with a reverse effect; see Table 4.

Table 4: The influence of the frequency of use and the impact of higher-level functionality upon the organization of the lower-level interaction.

	frequency of use	
	low	***high***
low	not critical	maximize efficiency
impact		
high	minimize risk	reduce impact

In chapter 5 each protocol was described in the same way. First, the information that is handled by the protocol was described. Next, it was indicated how that information is presented to users (I-feedback). After that, the functionality available to users for manipulating the information was introduced and it was specified how these functions are to be selected and how their execution is to be triggered. In the last part of protocol descriptions the E-feedback provided at that level to communicate these procedural aspects of the protocol to users was indicated; see Table 5.

Table 5: The protocol description format.

	what	**how**
internal organization	information	functionality & sub-protocols
communicated by	I-feedback	E-feedback

To summarize, it has been demonstrated that the LP framework can be used for the design of full-scale user interfaces. User-interface designers are confronted with more manageable design problems because the LP framework models user interfaces as a hierarchically organized collection of highly independent interaction protocols. It was demonstrated that extra levels of interaction need to be introduced when the number of higher-level functions exceeds the number of knobs and buttons available to select these functions and trigger their execution. It was also shown that design decisions relating to what lower-level protocols are introduced to support a higher-level protocol should take the frequency of use and the impact of the higher-level functionality into account. In addition, a format was introduced to systematically describe interaction protocols in which the internal organization is specified, as well as the way in which this internal organization is communicated to users.

7.3 More usable interfaces

First, we investigated the central LP concepts and how they relate to the usability of user interfaces. After that, it was demonstrated how full-scale user interfaces that are in line with the LP framework could be designed. The remaining question is whether this all leads to more usable interfaces.

To find an answer to this question, we tested three user interfaces with equivalent functionalities with users in a uniform way. The first user interface (called DAT recorder) was not designed according to the LP framework. The second one (called prototype) was designed in line with the LP framework. The third one (called DCC recorder) was systematically designed according to the framework. For each recorder, 10 subjects executed a set of 8 tasks three times. For each task in each of the three sessions, a task-completion score (indicating how well a task was executed) and a task-completion time (indicating how long it took the subjects to execute the task) were measured. These measurements made it possible to compare the usability of the recorders in terms of effectiveness by comparing task-completion scores, in terms of efficiency by comparing task-completion times and in terms of learnability by comparing how the completion scores and execution times evolved over the sessions. In addition, the degree of subjective satisfaction was derived from the subjects' answers to a questionnaire. The experiments showed a clear ranking in usability of the three digital audio recorders. The DCC recorder turned out to be

more usable than the prototype, while the prototype was more usable than the DAT recorder (see Figures 6 and 7). For more information, the reader is referred to chapters 3 and 6.

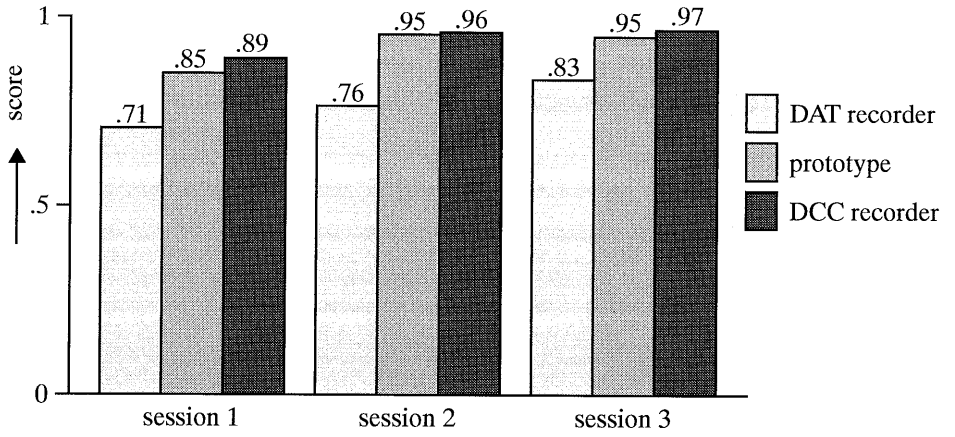


FIGURE 6. Effectiveness: the average task completion scores per session for the three recorders.

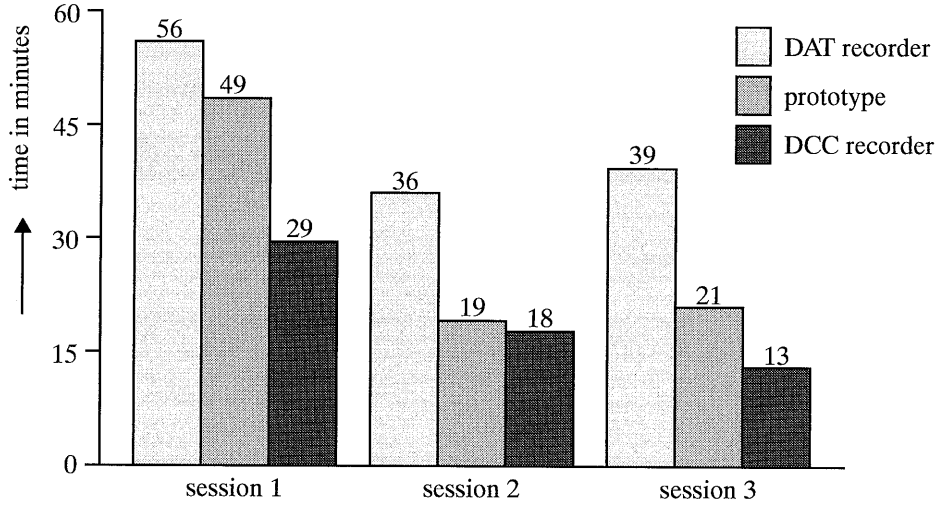


FIGURE 7. Efficiency: the average amount of time needed to complete the set of tasks per session for the three recorders.

The experiments showed that it is possible to develop more usable interfaces with the help of the LP framework. An important question is whether the usability differences found can be fully attributed to the Layered Protocols framework. The experiments did however not provide an answer to this question. The three user interfaces are very different although providing access to equivalent functionalities. Many decisions have to be taken during the design of a user interface. It was demonstrated that the LP framework provides guidance in taking some of those decisions. However, LP is far from complete in addressing all user-interface design issues. LP for example says nothing about the form of E- and I-feedback. It only states that E- and I-feedback should be made available to users at every level of interaction. Nor does it prescribe a precise organization of the interaction. It only states that the interaction organization should be hierarchical. Because of this, it cannot be concluded from the experiments that the usability differences are due completely to the guidance provided by the LP framework.

On the other hand, the results of the experiments do indicate that the LP framework addresses user-interface aspects that influence usability. In the case of both the prototype and the DCC recorder, the interaction between user and system evolved smoothly most of the time. This was indicated by the experimental results and the analyses of the video tapes and the usage logs confirmed this. Hence, we gather that most design decisions worked out as expected, including the ones guided by the LP framework.
During the experiment there were also critical incidents in which the interaction between subject and recorder was not as fluent as expected. These incidents were analyzed and it was attempted to explain the incidents in terms of the LP framework (e.g. lack of or insufficient E- or I-feedback). When a plausible explanation was found, the incident was attributed to incorrect exploitation of the LP framework. When no plausible explanation was found, it was decided that we had encountered an important usability issue that was not covered by the LP framework. It was found that most critical incidents could be explained using the LP framework.

To summarize, the experiments showed a clear ranking in the usability of the three digital audio recorders: the DCC recorder was more usable than the prototype, while the prototype was more usable than the DAT recorder. This shows that it is possible to develop more usable interfaces with the help of the LP framework. Although the usability differences do not have to stem fully from the LP model, the qualitative data of the experiments indicate that the LP framework indeed addresses aspects of user-system interaction that influence usability.

7.4 Conclusions

This thesis describes the development of a concise and operational version of the Layered Protocols framework emphasizing layered E- and I-feedback. It has provided a potential explanation for the role of E- and I-feedback in user-system interaction. It has also demonstrated how full-scale user interfaces can be systematically designed according to the framework. The hierarchical structure of the LP framework ensures that designers can more or less independently design the interaction protocols at the various levels, splitting

up the design problem into more manageable chunks. In addition, the results of the experiments show that structuring user-system interaction according to the Layered Protocols framework can lead to more usable interfaces for novice users. The experimental results indicate that the LP framework indeed addresses aspects of user-system interaction that influence usability. From all this we conclude that the Layered Protocols (LP) framework can indeed provide guidance in the design of full-scale easy-to-learn user interfaces for novice and occasional users.

7.5 Beyond this thesis

Having achieved this, what are the next issues to be resolved in order to make the Layered Protocols framework a firm basis for model-based user-interface design?

First of all, the LP framework needs a more formal definition. This is especially important with respect to making clear-cut decisions on whether a user interface is compliant with the framework. Currently, the framework is still too general. In principle, any user interface can be forced into the framework by introducing only two levels: the physical interaction level (dealing with the physical means for input and output like buttons and screens) and the application level consisting of one large, complex interaction protocol. This is possible because there are no explicit criteria for distinguishing between interaction layers. Futhermore, the framework only states that E- and I-feedback should be available for users at every level of interaction. It does not issue quality criteria for E- and I-feedback. A more formal definition of the model will help in formulating explicit criteria for distinguishing between interaction levels and for assessing the quality of E- and I-feedback.

A next step could be to design a series of user interfaces that vary systematically along the dimensions of the framework. By measuring the usability of those user-interface variants, it becomes possible to assess the impact of the dimensions indicated by the framework upon usability. A serious problem is how to measure the usability of those user interfaces. The method applied in this thesis measuring the overall usability of user interfaces is probably not sensitive enough to reveal differences in usability between user-interface variants. The solution probably lies in measuring usability per protocol. Measuring usability at various interaction levels is feasible because the framework treats protocols as user interfaces in their own right. By measuring the usability of protocols at the various levels, it may be possible to find usability differences at lower levels while there are no significant effects at higher levels. In order to reduce the amount of work and increase the accuracy of measurements of the usability of (sub-)protocols, automatic procedures should be developed to determine the performance-oriented aspects of usability (effectiveness, efficiency and learnability) on the basis of usage logs.

The LP framework has a compositional approach towards user interfaces. This triggers the question whether usability is compositional. Is it possible to predict the usability of a higher-level interaction protocol on the basis of the usability of its lower-level protocols? The model that serves as an example is GOMS (Card, Moran and Newell, 1983). By an-

alyzing the user-system interaction as it should evolve for achieving a set of representative tasks, an inventory can be made of the physical and cognitive processes expert users have to execute. By estimating the individual process-execution times, the overall task execution times can be computed and thus the efficiency with which users can fulfil their tasks (John, 1990). A drawback is that the GOMS model assumes that users have internalized the interaction procedures and do not spend time on learning them: the interaction should unfold according to a protocol mutually understood and exercised by the user and the system. This means that GOMS can only be applied when expert users are involved. Usability is reduced to efficiency. Effectiveness and learnability are no usability factors for GOMS. This is precisely what the LP framework could bring to the table. Would it be possible to estimate effectiveness, efficiency and learnability using the LP framework?

Along these lines, a third, and again more comprehensive, evaluation round of the Layered Protocols framework could be conducted, extending the results of the first two evaluation loops reported in this thesis. This third loop would again deepen our understanding of how the LP framework can provide guidance in the design of easy-to-learn user interfaces for novice and occasional users.

7.6 References

Card, S.K., Moran, T.P. & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

John, B.E.(1990). Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. In: *Proceedings of CHI, 1990*. NewYork, ACM.

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

Lewis, C.H. (1988). Why and How to Learn Why: Analysis-based Generalization of Procedures. *Cognitive Science*, 12, 211-256.

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

Payne, S. J. & Green, T. R. G. (1986). Task-action grammars: a model of the mental representation of task languages. *Human-Computer Interaction*, **2**, pp. 93-133.

Polson, P.G. & Lewis, C.H. (1990). Theory Based Design for Easily Learned Interfaces. *Human-Computer Interaction*, **5**, 191-220.

Powers, W.T. (1973). *Behavior: the control of perception*. Chicago, Illinois: Aldine Publishing Company.

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

Taylor, M.M. (1988b). Layered protocols for computer-human dialogue. II: Some practical issues, *International Journal of Man-Machine Studies*, **28**, pp. 219-257.

Taylor, M.M. (1992). *Principles for Intelligent Human-Computer Interaction, a tutorial on Layered Protocol Theory*. North York, Ontario: DCIEM report no. 93-32.

Summary

This thesis explores how the Layered Protocols (LP) framework can provide guidance in the design of full-scale easy-to-learn user interfaces for novice and occasional users.

The LP framework is evaluated in two rounds. In both rounds the framework is analyzed, a full-scale user interface is designed and the usability of that user interface for novice users is assessed.

In the first round the concepts introduced by the Layered Protocols framework are carefully studied. The role of expectations in user-system interaction is stressed, which is reflected by the introduction of the E-feedback and I-feedback concepts. After that a newly designed user interface for Digital Compact Cassette (DCC) recorders is described. The interaction is hierarchically structured as indicated by the LP framework. Next a description of a usability test is given in which the newly designed DCC recorder was tested with users. Its usability is compared with that of a commercially available Digital Audio Tape (DAT) recorder. These two devices offer largely the same functionality. The DCC recorder turned out to be more usable than the DAT recorder, due largely to effects that could be explained in LP concepts.

In the second round the Perceptual Control loop, integrated by Taylor in the LP framework, is studied in order to obtain a better understanding of why novice users behave as observed in the user evaluations. It is illustrated how behavior of novice users can be explained by perceptions created by the system that trigger some basic problem-solving strategies in users. After that a totally different user interface for DCC recorders is designed in a systematic way, structured in detail according to the LP framework. The hierarchical structure of the LP framework enables user interface designers to more or less independently develop the interaction protocols at the various levels, splitting up the design problem into more manageable pieces. By structuring protocol descriptions user-interface designers are invited to address the internal protocol organization as well as the way in which the protocol organization is communicated to users. This second DCC recorder was also tested with users. It proved to be more usable than both the DAT recorder and the first DCC recorder. The moments at which the interaction between user and recorder did not go smoothly could be explained in terms of the LP framework.

Overall this thesis describes the development of a concise and operational version of the Layered Protocols framework emphasizing layered E- and I-feedback. It provides a potential explanation for the role of E- and I-feedback in user-system interaction. It also demonstrates how full-scale user interfaces can be systematically designed and described according to the LP framework. In addition, the results of the experiments show that structuring user-system interaction according to the LP framework can lead to more usable interfaces for novice users. The experimental results indicate that the LP framework addresses interaction aspects that influence system usability. Therefore it is concluded that the Layered Protocols framework can indeed provide guidance in the design of full-scale easy-to-learn user interfaces for novice and occasional users.

Samenvatting (Dutch summary)

Dit proefschrift onderzoekt hoe het Layered Protocols (LP) raamwerk ondersteuning kan bieden bij het ontwerpen van gebruikersinterfaces van realistische omvang die eenvoudig te leren zijn voor beginnende en incidentele gebruikers.

Het LP raamwerk wordt in twee rondes geëvalueerd. In beide rondes wordt het raamwerk nader geanalyseerd, een gebruikersinterface van realistische omvang ontworpen en de bruikbaarheid van de interface voor beginnende gebruikers vastgesteld.

In de eerste ronde worden de concepten die door het Layered Protocols raamwerk worden geïntroduceerd zorgvuldig bestudeerd. De rol die verwachtingen spelen in de interactie tussen gebruiker en systeem wordt benadrukt, wat wordt weergegeven door de introductie van de E-feedback en I-feedback concepten. Daarna wordt een nieuw ontworpen gebruikersinterface voor Digital Compact Cassette (DCC) recorders beschreven. De interactie is hierarchisch gestructureerd, zoals door het LP raamwerk wordt aangegeven. Vervolgens wordt een beschrijving van een bruikbaarheidstest gegeven waarin de nieuw ontworpen DCC recorder werd getest door gebruikers. De bruikbaarheid van de interface wordt vergeleken met die van een commercieel verkrijgbare Digital Audio Tape (DAT) recorder. De beide apparaten bieden grotendeels dezelfde functionaliteit. De DCC recorder bleek beter bruikbaar dan de DAT recorder met name dankzij effecten die verklaard konden worden met behulp van LP concepten.

In de tweede ronde wordt de Perceptual Control loop die door Taylor in het LP raamwerk werd opgenomen, bestudeerd om beter te kunnen begrijpen waarom beginnende gebruikers het gedrag vertonen zoals dat in de gebruikerstesten geobserveerd werd. Aangegeven wordt hoe het gedrag van beginnende gebruikers verklaard kan worden vanuit de percepties die het systeem veroorzaakt en die leiden tot enige basisvormen van probleemoplossend gedrag bij gebruikers. Daarna wordt op systematische wijze een totaal verschillend gebruikersinterface ontworpen voor DCC recorders. De interface is tot in detail gestructureerd volgens het LP raamwerk. De hierarchische structuur van het LP raamwerk stelt ontwerpers van gebruikersinterfaces in staat om de verschillende interactieprotocollen op de verschillende lagen in de hierarchie min of meer onafhankelijk van elkaar te ontwikkelen, door het ontwerpprobleem op te splitsen in kleinere, meer hanteerbare deelproblemen. Door de beschrijvingen van protocollen te structureren worden interface-ontwerpers eraan herinnerd zowel de interne organisatie van protocollen te specificeren, als ook de wijze waarop deze interne organisatie aan gebruikers wordt duidelijk gemaakt. De tweede DCC recorder werd ook door gebruikers getest. Deze bleek beter bruikbaar dan zowel de DAT recorder als de eerste DCC recorder. De momenten waarop de interactie tussen gebruiker en recorder niet soepel verliep, konden worden verklaard aan de hand van het LP raamwerk.

Dit proefschrift beschrijft hiermee de ontwikkeling van een bondige en operationele versie van het Layered Protocols raamwerk waarin gelaagde E- en I-feedback wordt benadrukt. Het geeft een mogelijke verklaring van de rol van E- en I-feedback in de interactie

tussen gebruiker en systeem. Het proefschrift laat ook zien hoe gebruikersinterfaces van realistische omvang systematisch ontworpen en beschreven kunnen worden volgens het LP raamwerk. Bovendien tonen de resultaten van de experimenten aan dat het structureren van de interactie tussen gebruiker en systeem volgens het LP raamwerk kan leiden tot interfaces met een hoger niveau van bruikbaarheid voor beginnende gebruikers. De resultaten van de experimenten tonen aan dat het LP raamwerk interactie aspecten aangeeft die van invloed zijn op de bruikbaarheid van systemen. Daarom wordt de conclusie getrokken dat het Layered Protocols raamwerk inderdaad ondersteuning kan bieden bij het ontwerpen van gebruikersinterfaces van realistische omvang die eenvoudig te leren zijn voor beginnende en incidentele gebruikers.

Curriculum Vitae

May 31, 1961	Born in Leeuwarden, the Netherlands.
1979	Graduated at Lienward College (atheneum), Leeuwarden, the Netherlands.
1985	Ir. Electrical Engineering (cum laude), University of Twente, Enschede, the Netherlands
1985 - 1986	Lieutenant of the Royal Dutch Air Force, Royal Military Academy (Koninklijke Militaire Academie, KMA), Breda, the Netherlands.
1986 - 1995	Scientist, Philips Research Eindhoven and the Institute for Perception Research (IPO), Eindhoven, the Netherlands.
1995 - 1997	Visiting scientist, Philips Research Briarcliff, Briarcliff Manor, New York, United States of America.
1997 - present	Scientist, Philips Research Eindhoven and the Center for Research on User-System Interaction (IPO), Eindhoven, the Netherlands.

Stellingen

behorende bij het proefschrift

Layered Feedback in User-System Interaction

van Reinder Haakma

I

Het Layered Protocols (LP) raamwerk biedt ondersteuning bij het systematisch ontwerpen van gebruikersinterfaces van realistische omvang die eenvoudig te leren zijn voor beginnende en incidentele gebruikers.

Dit proefschrift.

II

Het LP raamwerk geeft interactie aspecten aan die van invloed zijn op de bruikbaarheid van systemen. Een hierarchische interactiestructuur gecombineerd met gelaagde E- en I-feedback zijn hiervan de belangrijkste.

Dit proefschrift.

III

De handelingen die gebruikers met systemen uitvoeren worden sterk bepaald door de verwachtingen die ze hebben over effecten ervan. Systemen kunnen met behulp van 'E-feedback' deze verwachtingen beinvloeden.

Dit proefschrift.

IV

De hierarchische structuur van het LP raamwerk stelt ontwerpers van gebruikersinterfaces in staat om het ontwerpprobleem op te splitsen in kleinere, meer hanteerbare deelproblemen. De verschillende interactieprotocollen op de verschillende lagen in de hierarchie kunnen min of meer onafhankelijk van elkaar ontwikkeld worden.

Dit proefschrift.

V

Het schoonmaken van de huidige koffieautomaten op de Technische Universiteit van Eindhoven doet E-feedback verdwijnen waardoor ze slechter bruikbaar worden voor beginnende gebruikers.

VI

Ten onrechte houden Cognitive Complexity Theory en de Task-Action Grammars in hun analyse van de complexiteit van de interactie tussen gebruiker en system geen rekening met de kwaliteit van de informatie die door systemen aan gebruikers gegeven wordt.

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

Payne, S. J. & Green, T. R. G. (1986). Task-action grammars: a model of the mental representation of task languages. *Human-Computer Interaction*, **2**, pp. 93-133.

VII

Puzzels, zoals de torens van Hanoi, hebben moeilijke gebruikersinterfaces.

Kotovsky, K., Hayes, J.R. & Simon, H. A. (1985). Why are some problems hard? Evidence from Tower of Hanoi. *Cognitive Psychology*, **17**, pp. 248-294.

VIII

Bruikbaarheid kan niet aan een apparaat worden afgezien. Dit maakt het moeilijk om een hogere prijs te vragen voor een beter bruikbaar apparaat.

IX

Een ergonoom die het gemak niet zoekt is lui.

X

Door de introductie van spraakherkenning in apparatuur wordt het waarheidsgehalte van het friese gezegde 'It is mei sizzen net te dwaan' bevestigd zowel als ondermijnd.