

# Feedback in Human-Computer Interaction — Characteristics and Recommendations

Karen Renaud\* and Richard Cooper  
University of Glasgow  
{karen,rich}@dcs.gla.ac.uk

May 2, 2000

## Abstract

The need for feedback is widely accepted, and the role of feedback in making things easier for the end-user is also no longer disputed. However, the nature of the feedback, and the manner of this provision is by no means agreed upon. Feedback is traditionally considered to be a communication of immediate system state to the end-user. This paper will argue the need for extending the concept to encompass the provision of *archival* feedback — information about past activity. We also argue the need for graphical, rather than textual feedback, and provide a list of desirable feedback features which should be provided by an application. The paper ends off by examining the means for this provision.

## 1 Introduction

The need for feedback in user-computer interaction has long been accepted. Many authors have published guidelines in an attempt to assist application developers in providing this feedback to the end-user [45, 3, 31]. In spite of this, adequate provision of feedback proves to be elusive. We have all used applications which do not supply adequate feedback, or applications that print out weird error messages, including numerical error codes. This makes it especially difficult to learn how new applications work — and to recover when something goes wrong. Computers have been integrated into all spheres and occupations, and the need for interpretability has become more urgent [1]. Andersen argues that the mode of operation and meaning of data should be understandable to all types of user — and that we should not expect the end-user to decipher cryptic messages, but rather encourage the use of the computer in supporting their task. The feedback mechanism is vital in achieving this goal.

Feedback has traditionally been seen as an responsive facility, one which simply signifies a confirmation on the part of the computer system. The view has been one of providing a corresponding action for every user action [45, 31], somewhat in the vein of every force having an equal and opposite force. We will argue for the extension of the feedback concept to encompass the following characteristics:

1. The provision of feedback about *both* immediate and past interaction.
2. The development of feedback's role in influencing the future behaviour of the user.
3. The extension of feedback to provide explanations of system actions, thus promoting user understanding of the computer application.
4. The promotion of the use of graphical rather than textual feedback mechanisms.

To complete the paper, a list of desirable feedback features will be given, and a means for the provision of such feedback will be proposed.

Section 2 describes the nature of feedback, and Section 3 addresses issues with respect to the motivation for feedback. The timeliness of feedback is discussed in Section 4, and the use to which the end-user will put the given feedback in Section 5. Since there is a need to pin down exactly what feedback should be provided, Section 6 will address this issue. The format of the feedback is explored in Section 7, and recommendations for mechanisms to provide the feedback are given in Section 8. Section 9 concludes the paper.

## 2 The Nature of Feedback

Feedback is a word widely used to denote different meanings in several academic areas including engineering, economics, biology, mathematical models or biological systems, formal logic, and social science [43]. The OED [48] defines feedback as:

---

\*On study leave from the University of South Africa (renaukv@alpha.unisa.ac.za)

1. The modification, adjustment or control of a process or system (as a social situation or a biological mechanism) by a result or effect of a process esp. by a difference between the desired and an actual result.
2. Information about the result of a process, experiment, etc.
3. A response.

Spink and Saracevic [49] argue that all academic perspectives have a basic concept of feedback as *involving a closed loop of causal influences, a loop of mutual or circular causality*. Feedback has a specific polarity — positive feedback reinforcing change in the same direction, negative feedback causing a change in the opposite direction, while homeostatic feedback maintains an equilibrium.

The human-computer interaction feedback perspective has an interest in “the exchange of information between participating agents through sets of channels, where each has the purpose of using the exchange to change the state itself or one or more others” [50]. The feedback thus concentrates on the method and type of interaction, the participants in the interaction, their purpose, and the interface between the human and the computer [49].

Shneiderman [45] defines feedback as *communication with a user resulting directly from the user’s action*. Pérez-Quñones and Sibert [38] point out that this definition does not cover communication from the system which notifies the user about the state of the system, or feedback about long-lived activities or transactions. They argue that feedback should be seen as serving a behavioural purpose in interaction, so that the computer fulfills the same role as a conversational participant. Suchman [52] argues that the immediacy of the reactions of computers of today, combined with the fact that such reactions are not random but considered (having been designed by a human programmer), lead people to consider the computer to be a purposeful social object — a conversational participant.

Participants in a conversation do not merely take turns, but in many ways collaborate in the conversation. The person doing the talking expects a level of feedback from the person being addressed, either in the form of nods, verbal affirmations, or facial expressions. These indicators are used so that the person doing the talking can determine whether the person being spoken to is receiving the message, and understanding it. A facial expression can convey a negative response to what a person is saying, or a positive response, indicating understanding. A blank expression, on the other hand, could indicate that the person being addressed is deaf, or does not understand the language of discourse. Thus the feedback can be seen to be either positive (when things are going smoothly) or negative (when the listener signals a problem), and the feedback will determine the future conversation [6].

This conversational feedback model fits in nicely with the interaction between humans and computers. Clark and Schaefer [12] proposed a model of collaborative contributions to conversation, and identified four states that the person being addressed can be in:

- not aware of being addressed;
- aware, but did not hear what was said;
- heard it, but did not understand;
- heard, and understood;

If we apply these principles to the human-computer conversation, it is logical to assume that the user will want to be able to identify these states in their conversational partner — the application — so that they know how to proceed. Suchman [52] likens the human-computer interaction to human-to-human conversation based on three points of similarity, with both being:

1. *reactive* — Computers react to user actions, meaning that the control of the human-interaction process is essentially in the hands of the user.
2. *linguistic* — The use of computers today is not a matter of pulling levers and pressing buttons, but rather of specifying operations and considering their results — exhibiting linguistic behaviour.
3. *opaque* — The general opacity of human participants in a conversation makes explanations about human intentions critical in understanding human action.

Suchman argues that the aforementioned reactive, linguistic and opaque properties of computers lead us, as users, to attribute intentions to the behaviour of the computer system. She further argues that, having drawn this conclusion, we then expect the computer to explain itself, and we expect it to behave in a rational way.

When humans communicate they often assume a shared background knowledge of the particular subject they are discussing. The speaker will have to gauge the listener’s understanding of the topic, and take steps to explain further if the speaker concludes that the listener does not have some knowledge they need in order to understand the conversation properly. The listener will also make assumptions about the speakers knowledge, and opinions as they speak. Suchman asserts that much of what is said often requires reference to other facts which are unspoken, but relevant to the conversation. Only by means of feedback can participants in a conversation detect faults in the understanding of what is said [17].

The success of human-computer conversation then, will depend on the human being able to gauge the “knowledge” of the application — and being able to supply the computer with those items it needs in order to continue the conversation successfully. Feedback is a valuable tool in the hands of an application developer, who needs to communicate the application’s “knowledge” and expectations to the user to facilitate the application’s role as conversational participant. In conclusion, perhaps the best definition of feedback would be *the communication of the state of the system, either as a response to user actions, to inform the user about the conversation state of the system as a conversation participant, or as a result of some noteworthy event of which the user needs to be apprised.*

### 3 Why give Feedback?

De Bono [14] maintains that it is often better to simplify a process than to train people to cope with the complexity. Feedback should be considered to be a way of simplifying the interaction between the user and the system. In motivating this “simplifying” feedback it is necessary to understand the nature of the interaction between the user and the computer.

One of the first attempts to model human interactive behaviour was done by Card, Moran and Newell [7], who proposed the GOMS (Goals, Operators, Methods and Selection) model. GOMS is a very good model for predicting timing properties, but not as good at accommodating the effects of human thought [15]. Norman’s [32] action-based approach, which analyses the interaction between humans and computers, identifies the stages of human activity shown on the left hand side of the following table, while the matching stages of conversational activity (since we are considering feedback needs with respect to conversational dialogue) are shown on the right:

Step	Stages of Human Activity	Conversational Stages
1	Establishing the goal	Establishing the goal
2	Forming the intention	Deciding what to say
3	Specifying the action sequence	Formulating the words in the mind
4	Executing the action	Saying the words
5	Perceiving the system state	Hearing the reply
6	Interpreting the state	Understanding the reply
7	Evaluating the system state	Interpreting what was said

During this activity, Norman identifies two gulfs that have to be bridged as a result of the difference between human goals (in psychological terms) and system states (expressed in physical terms). The two gulfs which need to be bridged to enable human use of a system are the gulfs of *execution* and *evaluation*. The gulf of execution represents the effort that the user has to make in order to translate their goals into action sequences which, when applied to the system, will achieve the goal. The gulf of evaluation represents the effort the user has to make to understand the state of the system as a result of their actions. Norman argues that these gulfs can be bridged from either direction. The system can narrow the gulf by constructing the interface with the needs of the user in mind. The user can bridge the gulf by creating plans, action sequences and interpretations of the system.

There are two different schools of thought with respect to the motivation behind user actions. The artificial intelligence branch of computer science is based on the existence of *underlying plans* influencing user actions [27]. The alternative to this rationalistic perspective is that action is inherently *situated* — with plans having a limited prescriptive affect on user actions [52]. Suchman’s *situated action* view is that users react to their circumstances, with an objective in mind, rather than slavishly following some set of plans.

The conversational model of user interaction, with respect to the current paradigm of recognition rather than recall [15], seems to lean towards the Suchman’s situated action perspective, rather than the plan-based mode of operation. Users behaving in this manner are even more dependent on the narrowing of the gulf of evaluation, since they react according to the way they interpret the state of the system. The plan-based approach suggests a fore-knowledge of the application’s user interface. The expert user may indeed have this knowledge, but the novice or occasional user would tend to react to the state of the application rather than act according to some set of plans.

The quality of the feedback provided by the system can go a long way towards narrowing the gulf of evaluation — in conversational terms, enabling an understanding of what was said. Feedback becomes very important when the system is prone to long response times, which often happens in distributed systems. A slow response could be indicative of an error, or simply a normal occurrence if the network is overloaded. Either way, the user needs to be fully informed about the reason for the delay. Feedback becomes critical in the case of system failure. Many systems simply stop functioning in the case of a system failure, and the user is left in the unenviable position of not knowing what has happened. The user will definitely be unsure about whether the activity that resulted in the failure is worth repeating or not.

Norman [33] argues that in any complex environment — for instance, a new application — one should always expect the unexpected. To deal with the unexpected, Norman concludes that continuous and informative feedback

is essential. Norman [52] mentions three different concepts which exist when the human computer interaction process is considered:

- The design model — the model held in the system designers mind of how the system should work.
- The user’s model — the mental model of the system, as built up by the user during user interaction with the system.
- The system image — which portrays the physical structure of the system.

As users use a system, they build up a user model of how the system works. In a conversation, the speaker is also able to gauge the knowledge of the listener during the course of a conversation — also building up a mental image of the thought processes and attitudes of the person being addressed. With respect to building this model for computer applications, users tend *not* to read manuals, wanting rather to find out for themselves how the system works [10]. They also tend to be impatient to get on with their task, and don’t want to spend hours being taught how to use a system [5]. This is in accordance with cognitive principles, which advocate “learning by doing” [2, 28]. Because of this, the user model will not be based on the design model, but rather on the system image. The designer thus has a difficult task in making this system image explicit, intelligible and constant [32].

Therefore, *feedback* is far superior to user manuals for helping the user to build up a correct internal model. The role of clear explanations in this process is vital [26]. Explanations of system actions can provide a sense of the underlying *purpose* of the system’s response to a user’s actions. Chan *et al* [11] have also shown that an active feedback system greatly improves user performance.

## 4 When must Feedback be Given?

The need for feedback has been argued in the previous section. Different authors attempt to provide guidelines to help developers to provide the right level of feedback. Waern [54] suggests that feedback should not be delayed, since the user needs it continuously to support their sequence of mental operations. Other researchers also urge that immediate and continuous feedback be provided [45, 3, 31]. Planas and Treurniet [40] have shown that continuous feedback reduces annoyance with respect to slow responses.

Marshall *et al* [29] refer to the difference between what they refer to as *required* (during execution of the task) versus *confirmatory* (at the end of the task) feedback. The former is required for more complex tasks, while the latter is suitable for simpler tasks, or tasks that have been mastered by a user who can be considered to be expert in that task.

Finally, it has been shown that feedback has an effect on the level to which a particular task is automated. When the feedback is immediately available, the user will be less likely to automate the task, and more likely to work in a controlled mode — making less errors. Thus, in complex tasks where the user should concentrate so as to notice exceptional circumstances which will require handling, the feedback should be more intense and available than for simple tasks which can be automated without risk [21]. To summarise, the rule seems to be: “Always provide feedback, for every action, and make sure it is completely unambiguous and informative”. Quite a tall order.

## 5 Use of Feedback

The previous sections have motivated feedback, and discussed issues pertaining to the timeliness of feedback provision. Before proceeding to further examination of feedback provision, and attempting to compile a list of desirable feedback features, we need to take a look at the *use* to which the user will put the feedback which is provided. The Oxford English Dictionary defines feedback as doing the following:

1. *signifying a response*. This serves to reassure, and confirm that their inputs have been accepted, and that the system is acting upon them.
2. *modifying the behaviour of the user*. If we once again consider the similarity of the human computer interaction in terms of a conversation, the feedback will serve to help the user to plan their following strategy, and help them to decide how to proceed. Without feedback, either negative, or positive, the user is left wondering whether to pursue their original course of action, or to veer to one side or another to accommodate some fault.
3. *promoting understanding*. The user needs to understand the system, and the effect that their inputs are having on the state of the system. Without a good understanding both of the present state, and the role they have played in bringing the application into that state, they cannot hope to proceed knowledgeably.

The traditional role of feedback in human-computer interaction is often seen exclusively as pertaining to the first use mentioned above. There is a need to widen that view to encompass the other uses, in order to genuinely provide good feedback.

Another use to which feedback can be put is that of giving an overview of some aspect of the system. The feedback could be used by some entity which needs to monitor performance of the application, by the system support person, or by the end user in providing them with some information not pertaining directly to the state of the system, but rather to other characteristics like performance, workload etc.

The extension of the feedback concept to include all the above-mentioned features will be addressed in the following section.

## 6 What is Good Feedback?

Some feedback is fairly standard, such as the highlighting of text to indicate selection, or the display of text and icon in reverse video to show that an icon has been selected. However, some feedback needs are not nearly as straightforward, and the developer may not have ready guidelines to be followed. An excellent example of this is the diverse treatment accorded to error management. Some applications will display an error message which requires some acknowledgement from the user before work can continue. Others simply generate a beep, and refuse to continue until the user provides a correct response, and yet others will simply display an enigmatic message and close the application. It is difficult to provide a general rule about the exact nature of the feedback since it is directly dependent on the nature of the task.

The feedback discussed in previous sections has referred to the communication of the “here and now” state of the system to the user. This feedback model is impoverished, and a strong case can be made to motivate the extension of the concept to encompass a historical perspective that would add a dimension to feedback hitherto unexplored.

### 6.1 The Nature of Feedback

It has been noted by various researchers that a discourse typically has an incremental quality about it [8, 25]. When people converse they often refer back to some part of their conversation in order to explain their present remarks. Dix [15] argues that it is difficult for users to manage and visualise this “sense of history” in their interaction with the computer, especially since the current interface is based more on recognition than recall. The user has no need to remember lists of commands, but simply chooses one from a menu. This historical need was also noted by Tweedie [53], who argues that past input and output should be linked so that all historical input and output relationships can be explored directly. This is echoed by Shneiderman [45]. Often the application’s only concession to a user’s need for this is the provision of an undo facility. Some tutorial-type, or visualisation applications supply the user with a log file containing previous explanations [13]. This does not link the explanations to user actions, though, and is therefore of limited assistance in providing feedback. If Norman’s stages of human activity are considered, the explanations only provide step 5 — the system state — whereas the user needs to understand the link between step 4 (their actions) and step 5 in order to correctly interpret the state of the system.

Another look at the conversational model will serve to illustrate this concept. If someone is recounting a conversation with a third party, the structure of the narrative will take the form: “She said ....., and then I said ...”. This is so that the person being addressed can understand the context of the narrator’s statements. It is no good only hearing one side of the conversation, and if the narrator chooses to do this, it will often lead to the listener being given a one-sided view, which is not conducive to understanding.

In addressing the question of what type of feedback is to be provided, it is therefore appropriate to consider the need for the portrayal of previous system states, so that the user can refer to it in order to understand the present state of the system. The user can rebuild, in their own minds, the situations which prompted their actions — reconstructing their train of thought, and reasoning processes.

The previous paragraph has motivated the need to keep a history of both the user’s actions, together with the system’s response. This type of information could be referred to as *archival feedback*, as opposed to *immediate feedback* which communicates the present state of the system. Such archival feedback provides the facility often used in conversation when a person refers to a previous statement, and builds on it. In the light of this discussion, good feedback would thus involve giving the user both immediate and archival feedback.

The previous section has discussed the use to which feedback will be put. We can now bring these two concepts together, by marrying the concepts of use of feedback, with either immediate or archival feedback, as follows:

1. signaling a response — satisfied by immediate feedback;
2. changing behaviour, and promoting understanding — satisfied by both immediate and archival feedback. Immediate feedback allows the user to judge the immediate state of the system, while archival feedback would allow them to obtain a deeper understanding of how the system arrived at that state over a period of time;

This section could be made very long and become arduous to read, but instead will consist of merely one example of the type of feedback provided in well-known applications. When using Internet Explorer, you will notice that the buttons at the top of the Internet Explorer window include a “Print” button, which provides no feedback. I am often unsure about whether I have printed a document or not — especially when my train of thought has been interrupted. Since the printer is in another room, the only way to make sure is to get up and check whether the document has come out of the printer. It would be relatively simple to display a graphical feature which would change when the displayed page had been printed already. This would leave me in no doubt about whether my command had been acknowledged by the system, and whether the page had been printed or not.

Why is feedback provision often so inadequate? It could be because the application programmer is expected to provide for the feedback needs of at least three completely different types of users (end-user, programmer and system support) — often without guidelines, trusting only their own instincts. A variety of users will use any application during its lifetime. The first is the programmer, the next is the end-user, and then there is a system support person supplying assistance to the end-user. Each needs a different type or flavour of feedback. Many applications in use today are evidence of the variability of feedback provision by different programmers. Some possible reasons for this variability will be briefly discussed:

1. The programmer belongs to the world of technology, and finds it hard to conceive of users who do not have this understanding. The system developer brings a store of background knowledge to their task, and they have to assume a certain taken-for-granted knowledge in the end-user [19]. Assumptions of this nature, about people not known to the developer, are bound to be inaccurate. Consequently, it is extremely difficult, especially for a programmer without formal training in human-computer interaction, to provide feedback at the level required by the user. Since there is a shortage of programmers with specific training in human-computer interaction [30, 51], it is realistic to expect that most applications will fall short of the ideal level of feedback.
2. Many errors occur at a depth in the system where there is no awareness of the current state of the dialogue with the user. Thus the program from which the reporting emanates typically has no idea of the context from which the user needs to be relocated so that they can respond to the error.
3. Grudin [23] published a paper which looked at the issue of technologies where one person did work for which another person would reap the benefits. This was coined by Norman [34] as *Grudin’s Law* :

“When those who benefit are not those who do the work, then the technology is likely to fail, or, at least, be subverted.”

The programmer achieves little benefit from providing the right level of reporting for other types of user. Indeed, some organisations actually *profit* from software which provides inadequate feedback — by requiring users to pay for advice on using their systems.

It is clear from the previous discussion that feedback is vital, and that it is often neglected by application developers, to the detriment of end-users. The published guidelines do not seem to go far enough in providing a clear path for developers to follow in providing the necessary feedback. The following section will attempt to remedy this by consolidating the work by researchers in this field into a list of desirable feedback features.

### 6.3 List of Desirable Feedback Features

It would be useful to be able to have some sort of list of requirements, a milestone to measure actual system feedback against what could, or should be provided. Bannon [4] underlines the need for research results which have an applicability to design, rather than delivering only tools for post-factum analysis. We have argued for the provision of both immediate and archival feedback. The features listed below have been chosen to meet both those needs. A list of desirable features would include the following:

- Immediate Feedback:
  1. Keep the user informed about system state [44], i.e. whether the system [20]:
    - has received their request;
    - is working on it; or
    - has a problem.
  2. Explain unusual occurrences and errors [52]. Provide context sensitive assistance [21].
  3. Make visible what would be invisible, and improve the user’s feeling of control [35]. Give each action an obvious and immediate effect [45]. In addition, the feedback should be structured in such a way that the user is left in no doubt as to which particular action it refers to [24, 21], with Nielsen [31] advising that the user’s input should be rephrased and returned to them to indicate what the system has done as a result of it.

1. Mental aids to help users remember things [45, 35]. User have severe limits with respect to memory, as illustrated by the following examples [37]:
  - Users sometimes forget what they have done, especially if they are interrupted during a processing session.
  - Users often do not detect their errors. Often the user is vaguely aware that something has gone wrong, but has no idea how this occurred.
  - Difficulties are often experienced in holding recently experienced information until needed.
  - Users experience problems retaining information retrieved from long-term memory — such as remembering where they are in a plan of action (if they are operating in that mode).
2. Inter-referential feedback. Draper [16] argues the importance of a mutual reference between user input and application reaction so that the previous parts of the user-machine dialogue can be referred to. If we accept the situated action perspective, we see the user as operating with an objective in mind, rather than following a rigid plan of action. The user is therefore responding to the system’s state as they use the application, and the reconstruction of this state is extremely important in enabling the user to rebuild the circumstances (in their own minds) that prompted their action in the first place.

It is unusual for any system to provide this standard of feedback. In providing the feedback, there are difficulties which beset us, as described in the next section.

## 6.4 Provisos

Humans are diverse and wondrous creatures, and their very versatility makes the provision of feedback, along with other features of human computer interaction, anything but straightforward. Shneiderman [45] discusses the factors which should be kept in mind, like physical abilities and physical workspaces, cognitive and perceptual abilities, personality differences, cultural and international diversity, and the needs of the disabled, and the elderly.

It is difficult, if not impossible, for an application to provide feedback which is customisable by a specific user, to their needs, as shown above.

## 6.5 Consolidation

So far, this paper has argued the necessity of feedback, and given guidelines about how it should be provided. The need for both immediate and archival feedback has been argued, and desirable feedback features have been given.

It has been pointed out that feedback should be tailored towards the needs of the end-user, and that it would be a difficult task for applications to provide for all the possibilities mentioned in the previous section. Thus, we can conclude that the provision of feedback is not easily achieved.

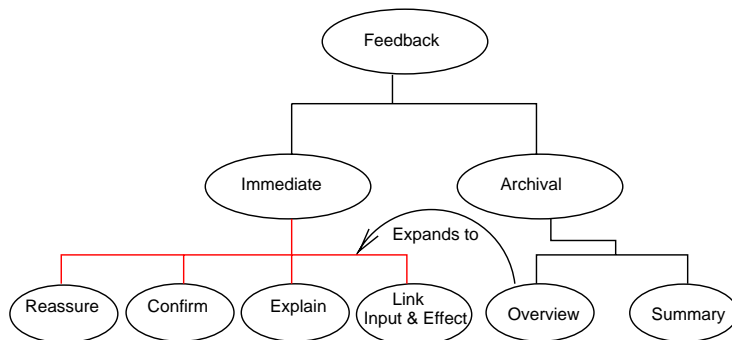


Figure 1: A Classification of Feedback

In order to synthesise the recommendations cited in this paper, a classification of the nature of feedback has been constructed, and is illustrated in Figure 1.

## 7 Feedback Format

The previous sections have argued the necessity of feedback, and discussed the type of feedback, and the difficulties inherent in feedback provision. This section will address the issue of the format of the feedback.

The issue to be resolved is whether feedback should be given in textual or graphical format. In human discourse many different communication channels are used to provide feedback. Apart from utterances, people also use gestures, gaze and body stance to communicate their understanding of what is being said [17]. A feedback model based only on textual descriptions will therefore not exploit the multiple possibilities available to us in providing feedback to the user.

Shneiderman advises that a feedback display should be consistent — using the same colours, formats, capitalisation etc. so that users will know what to expect, and that feedback should always be given where it is easily detected [45]. This can apply equally to textual or graphical feedback. However, there is a body of research which points unhesitatingly towards the advisability of graphical feedback.

Norman advises that sound and graphics should be investigated [35]. Faulkner, too, advises that feedback be presented in a graphical format, and that all feedback messages should be clear and unequivocal [18]. Phillips [39] argues that visual imagery is superior to verbal representation in aiding memory and thinking. Gardiner [21] agrees, saying that recall is better for dynamically interacting items than for items stored in isolation. She avers that recall is further improved if items are presented pictorially, rather than textually.

From a cognitive perspective, graphical feedback would be far more helpful, since users have particular strengths which can be utilised by non-textual feedback mechanisms like processing visual information rapidly, coordinating multiple sources of information, and making inferences about concepts or rules from past experiences [37].

Since the user's interaction with modern computer systems is essentially based on recognition, rather than recall, and is intensely visual, it would be less than optimal to try to describe a set of actions in a textual format. The representation chosen for a particular set of data will indeed make a difference [47] — some representations allowing users to perceive one type of pattern in the data, while another reveals something totally different. We should therefore explore possibilities for portraying feedback in a graphical format.

For instance, to assist the user in rebuilding their context, listed as an archival feedback requirement in Section 6.3, we would recommend that the best way of facilitating this is to provide an *action replay* of the user's interaction with the application — up to the point where they were interrupted. Since their original circumstances were established based on the recognition of certain windows on the screen, the reconstruction of this state should ideally be facilitated by visual means as well.

## 8 Who should Provide the Feedback?

This may seem an unusual question to pose — since most developers would feel that the obvious person to provide the required feedback should be the programmer of the end-user application. However, there are at least three different ways of providing this feedback:

1. Within the application — Programmers could provide this feedback while implementing the application. This is the best solution from the user's point of view, since the application programmer is best situated to understand the inner functioning of the application, and the feedback can be perfectly pitched towards the user's interaction with the application. The previous discussion has argued that this is unlikely to be done properly, but even if it were provided, there is the issue of duplication since each application needs a core of generic feedback-producing activities. The programmer could be assisted in this task by the use of a generic feedback Application Programmer Interface (API), but that would still require the programmer to put extensive effort into this aspect of their program development.
2. By means of a collaborative agent — Rich and Sidner [42] have developed a collaborative interface agent which maintains the history and context of the interaction between the user and the application. The agent interacts directly with the application, and with the user. It then maintains a history based on interaction with the user, and observation of user actions. Two windows are used to facilitate the visualisation of the user's communication with the agent, and the agent's communication with the application. The communication is based on an artificial language developed by Sidner [46]. They provide explanations of the system activity in a textual format, but do not explicitly show the link between user actions and system activities. They also require the application to provide specific *hooks* to facilitate the collaborative agent's function.
3. Independently of the application —
  - By means of an online help system — Online help only fulfills an explanatory role, and has difficulty in tendering context sensitive feedback. The issue of dynamic feedback is not catered for by online help. Another factor that makes this option untenable is the nature of human cognition, as mentioned in Section 3, making manuals a suboptimal way of assisting users.
  - By means of application tracking — This option tracks the application activity in order to provide feedback in a generic manner not specific to the particular application. This enables reuse of feedback facilities and makes the application programmer's task much simpler. Section 6.3 set out a list of desirable feedback features, and argued that very few applications currently offered these features. Section 6.4 pointed out the difficulty associated with making feedback suit the needs of different types

of users. These problems can be solved by using a generic feedback enhancing framework. HERCULE is an example of such a framework. HERCULE tracks application activity and provides both context sensitive immediate feedback, archival feedback, and overview functions [41].

Providing feedback from within the application is, as stated, potentially the best. It is also the approach currently being used, and is obviously not working very well. The second method has some potential, although the present implementation communicates actions in a textual format, which is somewhat limited. Providing feedback independently by means of online help is a method with glaring insufficiencies, and does not merit consideration as a total solution. The generic application tracking option is therefore the recommended one. While this method cannot give feedback with respect to internal application functioning, it can capture evidence of the application's interaction with the environment — both with the user and with the rest of the system, and provide comprehensive feedback based on that information. This technique is especially useful for portraying archival feedback, and can also be of limited use in providing immediate feedback.

## 9 Conclusion

This paper has examined aspects of feedback from a human-computer interaction point of view in some detail. The issues of why feedback should be provided, when it should be given, and the use a user makes of the feedback have been addressed. We have argued that feedback should have both immediate and archival features. We have also described the difficulties inherent in the provision of this feedback, and set out some desirable feedback features. The means for the provision of feedback have been enumerated and a recommendation made.

## Acknowledgment

We acknowledge the valuable contributions of Phil Gray. This research is supported by a scholarship from the Association of Commonwealth Universities and a grant from the Foundation for Research and Development in South Africa, and the University of South Africa. Karen is currently on leave of absence from the University of South Africa and would like to acknowledge their magnanimity in allowing her this extended period of absence.

## References

- [1] P. B. Andersen. *A Theory of Computer Semiotics*. Cambridge Series on Human-Computer Interaction. Cambridge University Press, Cambridge, 1990.
- [2] J. R. Anderson. *The Architecture of Cognition*. Harvard University Press, Cambridge, Massachusetts, 1983.
- [3] *Human Interface Guidelines: The Apple Desktop Interface*. Addison-Wesley, Reading, Massachusetts, 1987. Apple Computer Inc.
- [4] L. Bannon. From cognitive science to cooperative design. In N. O. Finneman, editor, *Theories and Technologies of the Knowledge Society*, pages 33–59. 1989.
- [5] N. Borenstein. *Programming as if People Mattered*. Princeton University Press, Princeton, New Jersey, 1991.
- [6] S. E. Brennan and E. A. Hulteen. Interaction and feedback in a spoken language system. In *AAAI Technical Report FS-93-05*, pages 1–6, 1993.
- [7] S. Card, T. Moran, and A. Newell. *Applied Information-Processing Psychology*. Erlbaum Associates, Hillsdale, NJ, 1983.
- [8] G. Carenini and J. D. Moore. Generating explanations in context. In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, Session 6: User Support, pages 175–182, 1993.
- [9] J. M. Carroll, editor. *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*. MIT Press, Cambridge, MA, 1987.
- [10] J. M. Carroll and M. B. Rosson. The paradox of the active user. In [9], chapter 5, pages 80–111. MIT Press, 1987.
- [11] H. C. Chan, K. K. Wei, and K. L. Siau. The effect of a database feedback system on user performance. *Behaviour and Information Technology*, 14(3):152–62, 1995.
- [12] H. H. Clark and E. F. Schaefer. Collaborating on contributions to conversations. *Language and Cognitive Processes*, 2:1–23, 1987.
- [13] N. Dahlback, A. Jonsson, and L. Ahrenberg. Wizard of oz studies – why and how. In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, Session 7: Design & Evaluation, pages 193–200, 1993.
- [14] E. de Bono. *Simplicity*. Penguin, London, 1998.
- [15] A. J. Dix. Closing the loop: modelling action, perception and information. In M. F. C. T. Catarci, S. Levialdi, and G. Santucci, editors, *AVI'96 - Advanced Visual Interfaces*, pages 20–28. ACM Press, 1991.
- [16] S. Draper. Display managers as the basis for user-machine communication. In D. A. Norman and S. W. Draper, editors, [36], chapter 16, pages 339–352. Lawrence Erlbaum Associates, Publishers, Hilldale, New Jersey, 1986.
- [17] F. L. Engel and R. Haakma. Expectations and feedback in user-system communication. *International Journal of Man-Machine Studies*, 39(3):427–452, 1993.
- [18] C. Faulkner. *The Essence of Human Computer Interaction*. Prentice Hall, London, 1998.
- [19] B. H. Fernando Flores, Michael Graves and T. Winograd. Computer systems and the design of organizational interaction. *ACM Transactions on Information Systems*, 6(2):153–172, April 1988.

- [20] J. D. Foley and A. van Dam. *Fundamentals of Interactive Computer Graphics*. Addison-Wesley, Reading, Mass. London, 1982.
- [21] M. M. Gardiner. Principles from the psychology of memory. In [22], chapter 5, pages 119–162. John Wiley & Sons, 1987. Part II. Episodic Memory.
- [22] M. M. Gardiner and B. Christie, editors. *Applying Cognitive Psychology to User Interface Design*, Chichester, 1987. John Wiley & Sons.
- [23] J. Grudin. Social evaluation of the user interface: who does the work and who gets the benefit. In H.-J. Bullinger and B. Shackel, editors, *INTERACT 1987. IFIP Conference on Human-Computer Interaction.*, Stuttgart, Germany, 1987. IFIP, Elsevier Science Publishers B.V.
- [24] N. Hammond. Principles from the psychology of skill acquisition. In [22], chapter 6, pages 163–188. John Wiley & Sons, 1987.
- [25] B. Lemaire and J. Moore. An improved interface for tutorial dialogues: Browsing a visual dialogue history. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, volume 2 of *PAPER ABSTRACTS: Accessing and Exploring Information*, page 200, 1994.
- [26] C. Lewis. Understanding what's happening in system interactions. In D. A. Norman and S. W. Draper, editors, [36], chapter 8, pages 171–186. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, 1986.
- [27] C. Lueg. Supporting situated actions in high volume conversational data situations. In *Proceedings of ACM CHI 98 Conference on Human Factors in Computing Systems*, volume 1 of *Monitoring the Complexity of Real Users*, pages 472–479, 1998.
- [28] K. Manktelow. Principles from the psychology of thinking and mental models. In [22], chapter 4. John Wiley & Sons, 1987.
- [29] C. Marshall, C. Nelson, and M. M. Gardiner. Design guidelines. In [22], chapter 8, pages 221–278. John Wiley & Sons, 1987.
- [30] T. Merridenard and J. Bird. Filling the gap. In *Management Today*. June 1999. Produced for Microsoft.
- [31] J. Nielsen. *Usability Engineering*. AP Professional, Boston, 1993.
- [32] D. Norman. Cognitive engineering. In D. A. Norman and S. W. Draper, editors, [36], chapter 3, pages 31–62. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, 1986.
- [33] D. Norman. The “problem” of automation: Inappropriate feedback and interaction, not “overautomation”. Technical Report ICS Report 8904, Institute for Cognitive Science, University of California, San Diego, La Jolla, California, 92093, 1989.
- [34] D. A. Norman. *Things That Make Us Smart : Defending Human Attributes in the Age of the Machine*. Addison Wesley Publishing Company, 1994.
- [35] D. A. Norman. *The design of everyday things*. MIT Press, London, England, 98.
- [36] D. A. Norman and S. W. Draper, editors. *User Centred System Design. New Perspectives on Human-Computer Interaction*. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, 1986.
- [37] J. R. Olsen. Cognitive analysis of people's use of software. In [9], chapter 10, pages 260–293. MIT Press, 1987.
- [38] M. A. Perez-Quinones and J. L. Sibert. Negotiating user-initiated cancellation and interruption requests. In *Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems*, volume 2 of *SHORT PAPERS: Models*, pages 267–268, 1996.
- [39] R. J. Phillips. Computer graphics as a memory aid and a thinking aid. *Journal of Computer Assisted Learning*, 2:37–44, 1986.
- [40] M. A. Planas and W. C. Treurniet. The Effects of Feedback During Delays in Simulated Teletext Reception. *Behaviour and Information Technology*, 7(2):183–191, 1988.
- [41] K. V. Renaud. HERCULE: Non-invasively Tracking Java Component-Based Application Activity. In *14th European Conference on Object-Oriented Programming. ECOOP 2000.*, Sophia Antipolis and Cannes, France., 12 – 16 June 1998.
- [42] C. Rich and C. L. Sidner. Segmented interaction history in a collaborative interface agent. In *Proceedings of the 1997 International Conference on Intelligent User Interfaces, Planning Based Approaches*, pages 23–30, 1997.
- [43] G. P. Richardson. *Feedback Thought in Social Science and Systems Theory*. University of Pennsylvania Press, Philadelphia, 1991.
- [44] P. A. Savage-Knepshield and N. J. Belkin. Interaction in Information Retrieval: Trends over Time. *Journal of the American Society for Information Science*, 50(12):1067–1082, 1999.
- [45] B. Shneiderman. *Designing the User Interface*. Addison-Wesley, Reading, Massachusetts, 1998.
- [46] C. L. Sidner. An artificial discourse language for collaborative negotiation. In B. Hayes-Roth and R. Korf, editors, *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 814–819, Menlo Park, California, 1994. American Association for Artificial Intelligence, AAAI Press.
- [47] H. A. Simon. *The Sciences of the Artificial*. The M.I.T Press, Cambridge, Massachusetts, 1969.
- [48] J. A. Simpson and E. S. C. Weiner, editors. *Oxford English Dictionary*. Clarendon Press, Oxford, second edition, 1989.
- [49] A. Spink and T. Saracevic. Human-computer interaction in information retrieval: Nature and manifestations of feedback. *Interacting with Computers*, 10(3):249–267, 1998.
- [50] G. Storrs. A conceptualization of multiparty interaction. *Interacting with Computers*, 6(2):173–189, 1994.
- [51] W. Strigel. What's the problem: Labor shortage or industry practices. *IEEE Software*, 16(3):52–54, May/June 1999.
- [52] L. Suchman. *Plans and Situated Actions*. Cambridge University Press, Cambridge, 1987.
- [53] L. Tweedie. Characterizing interactive externalizations. In *Proceedings of ACM CHI 97 Conference on Human Factors in Computing Systems*, volume 1 of *PAPERS: Information Structures*, pages 375–382, 1997.
- [54] Y. Waern. *Cognitive Aspects of Computer Supported Tasks*. John Wiley & Sons, Chichester, 1989.