Error Recovery Representations in Interactive System Development

Francis Jambon

Third Annual ERCIM Workshop on "User Interfaces for All", Obernai, France, 3-4 November 1997, p. 177-182

Error Recovery Representations in Interactive System Development

Francis JAMBON

LISI / ENSMA, BP 109
F-86960 Futuroscope cedex, France
E-mail: jambon@ensma.fr

Abstract. This paper deals with human error resistance. In the first part of it, a short state-of-the-art of human error resistance, i.e., error prevention and error correction will be presented. Then, error correction, which is usually divided into three sequential tasks: detection, explanation, and recovery will be described.

The second part of the paper will put emphasis on error recovery, which is our main object of study. First and foremost, through an example, we will see what makes the distinction between forward and backward error recovery, i.e., how the system is able to restore its previous state after an error occurrence. In addition to it, the three different kinds of backward error recovery – undo, cancel, and stop – will be illustrated.

Then, the limits of the current distinction between forward and backward error recovery will be highlighted. As a consequence, the paper proposes a new representation of error recovery based on two dimensions: *error additional cost* and *system state degradation*. In the context of time-critical systems, another dimension will be introduced: *time*.

1. INTRODUCTION

Although human errors are often observed in human-computer interaction, they are usually overlooked or ignored by most interface designers. Poor support for human error management in interactive system development may be an explanation of the designer's attitude. In this paper, taking into account designers' point of view, our aim is to support design for error recovery practices in interactive system development thanks to models. We all know that novice, very young, or elderly people make more often errors than expert users. So, in order to create "User Interfaces for All", designers must include good human error resistance strategies into the user interface life cycle. In this article, we propose a new distinction between error recovery methods in order to help practitioners to choose between recovery alternatives.

2. RELATED WORK

2.1. Human Error

Human Error has been widely studied from the cognitive science perspective (Leplat, 1985; Rasmussen, 1986; Reason, 1990) as well as from the applied science perspective (Swain & Guttmann, 1983; Nicolet, et al., 1990). The definition of "human error" is an important topic, but cannot be reviewed here. For an overview on this topic, we suggest reading Reason (1990).

One of the most significant result of human error studies – from the designer's point of view – is the evaluation of the Human Error Probability. As an example, the THERP method (Swain & Guttmann, 1983) focuses on nuclear power plant control rooms. More recent works suggest that – extended – interface specification notations can be used to detect a breakdown during human-computer interaction (Johnson & Gray, 1996).

With these results, designers can evaluate human error rates and their consequences on the system state, at the very first step of the design process. So, at this stage, human error resistance strategies can be defined in order to prevent or limit the consequences of human errors.

2.2. Error Resistance Strategies

Error resistance can be achieved by means of two strategies: *error prevention* and *error correction* (Lewis & Norman, 1986). *Error prevention* can result from forcing functions (Norman, 1990), operator's selection, or training. But error prevention will never get rid of all

human error occurrences. That's why *error correction* – sometimes called error management – has to be actually supported in interactive systems.

Human error resistance is mostly supported – when that is the case – by user interface designers through prevention. Designers ought to follow the prevention strategy, for self-evident reasons. Unfortunately, they usually deal with human error only with the prevention strategy. And yet, scientific published works (Lenman & Robert, 1994a), as well as aircraft accident reports (Ministère de l'équipement des transports et du tourisme, 1994)[1] stress the need for correction. In the former report, the authors criticize the fact that the human-system interface has given the crew too little chance to correct their disastrous error. Now, let's explore error correction.

2.3. Error Correction

As an operator, correcting an error or a system failure, can be achieved by following three sequential tasks: *Error detection*, i.e., the user needs to know that an error occurred ; *Error explanation*, i.e., the user must understand the nature of the error – do note that for the "undo" function, the user does not always need to know that ; *Error recovery*, i.e., the user has to counteract the effects of the error (Lenman & Robert, 1994a). Some systems have embedded auto-correction features (Lewis & Norman, 1986): in such systems, the error correction is achieved by the system which is in charge of the corrections tasks – detection, explanation, and recovery.

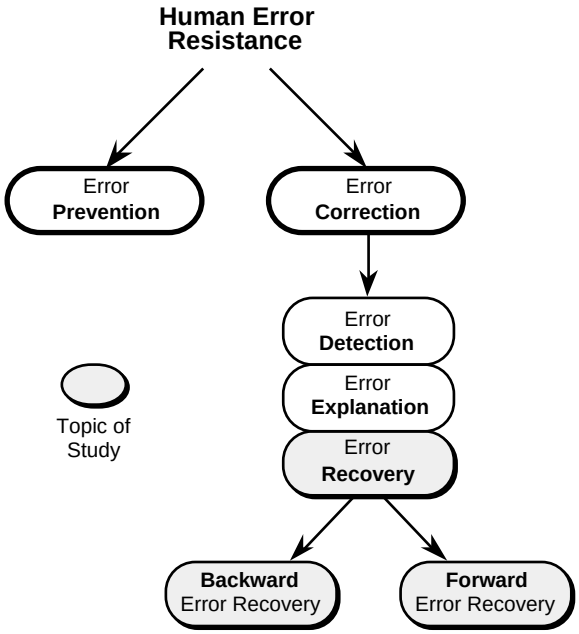

Figure 1 : Strategies to deal with Human Errors

Error detection and explanation are the first steps of error correction. They are primarily related to the interface presentation. In this paper, we focus on the recovery part of the error correction which involves the definition of the recovery procedures. Although error detection and explanation are not our topic, we assume that they are compulsory in a perspective of design for error. For self-evident reasons, error recovery is void if the user cannot neither detect nor understand the error ! Figure 1 summarizes the strategies that will help us to overcome human error and highlights our topic of study.

3. FORWARD VERSUS BACKWARD ERROR RECOVERY

Dix, Finlay, Abowd, & Beale (Dix, et al., 1993) make a distinction between backward and forward error recovery. This distinction is now commonly used in interactive system development. However, we suggest that this distinction is not as useful as it could be from the designers' point of view. So we propose our own complementary taxonomy.

3.1. Backward error recovery

Backward error recovery is an attempt to restore the system state after an error has occurred. Backward recovery can be considered as the only real "recovery" function, since the unexpected effects of error are totally removed. Backward error recovery can be seen as a function to go back in time. As a result, novice users usually use it as a fail-safe learning method. According to Yang (Yang, 1992) – in (Lenman & Robert, 1994a) – there are three kinds of backward error recovery commands: *undo*, *cancel*, and *stop*.

• The *undo* function is the most famous one, and most editors – for texts or graphics – implements an undo function. Unfortunately, the undo function is also the most complex one. Designers have to do with the presentation, the granularity, the scope, and the range of the undo

[1] Non French-speaking readers can find an english abstract of this report and some related discussions in the "Risks" forum archives: <http://catless.ncl.ac.uk/Risks/13.23.html#subj8>.

function. Moreover, current implementations of the undo function seem to fall short of user expectations (Lenman & Robert, 1994b).

• The *stop* function is used to terminate the process under execution. For example, a user can make the choice to stop a long printing command when he realizes that many users are waiting for the printer. However, the *stop* function is not always implemented as a pure backward recovery: in the former example, the user can stop the printing command while the printer is working, so, some pages may have been printed uselessly.

• The *cancel* function is used to abandon commands under specification. For example, a user can cancel the typing of an e-mail message if the addressee has just entered the user's office. However, as for the *undo* function, the *cancel* function has to deal with its scope: the user must know which command(s) are concerned by the recovery.

3.2. Forward error recovery

In forward error recovery, the user has to execute unexpected tasks to recover the fault. Usually the final result – the system state – is non-optimal. For example, if you break a dish plate, you have to use glue to recover your error. Of course, the final dish plate is not as nice as the unbroken one. In civil aviation, a lot of emergency procedures (Airbus Industrie, 1992) are forward recovery procedures. In these procedures the "error" is a system failure. The result is generally a non-optimal state of the aircraft: after an engine fire, the latter can remain inoperative for the flight.

Forward recovery is commonly the only way to recover from technical failure or human error in critical systems like nuclear power plants, chemical plants, aircraft, vessels, etc. In these systems, side-effects of many actions cannot be easily removed. As Lenman & Robert (1994a) say "Forward error recovery is an important topic, and more research is needed concerning these questions".

3.3. Limits of this classification

Dix et al. (1993) use the example of an house of cards to show the difference between backward and forward error recovery: "for example, in building a house of cards, you might sneeze whilst placing a card on the seventh level, but cannot undo the effect of your misfortune except by rebuilding".

3.3.a. Erroneous task granularity

Dix et al. suppose that you cannot undo the fall of your house of card after placing a wrong card. Obviously Dix et al. consider that the erroneous task is placing *one* card on the house. But that's not so obvious, if we consider that the real task is building an eight level house of cards. You may place a card in a wrong way, and then, the house falls down. Now collect all the fallen cards: that's a backward error recovery because you have undone the effect of your error. The system is exactly in the state before the beginning of your task "building the house of cards". Rebuilding the house of cards is now considered as a new task occurring after the recovery. So, the difference between backward and forward recovery is relative to the granularity of the erroneous task.

3.3.b. Additional cost of error

It is assumed that backward error recovery – undo, cancel, & stop – is the best way for a user to recover from an unexpected error. Clearly, pressing the "undo" button after a typing error is easy. Yet to achieve this goal, the user must retype the right letter, word, paragraph, or why not, the document. The cost of an error is clearly the cost of the correction task, as well as the cost of the correctly executed task. So, many users may prefer to use forward error recovery in order to move the system to a non-optimal state but at a lower cost. This is a critical topic from the designer's point of view.

3.4. Representations of Error Recovery

In order to avoid confusion between backward and forward error recovery, we propose a new taxonomy of error recovery. This taxonomy is based on two dimensions: Error additional cost and system state degradation.

3.4.a. Error additional cost

The first dimension, the *error additional cost*, represents the cost of the tasks the user has to perform in addition to the normal error-free task in order to manage the error. In other words, the *error additional cost* is the difference between the error-free task and the addition of the erroneous task, the recovery, and the new task performed to achieve the goal. This cost can be evaluated as the time needed to ensure the completion of the tasks, as well as the cognitive or the conative (Dorwell & Long, 1989) cost of the additional tasks. It represents, for the user's point of view, the extra cost or the disappointment generated by his error (or system failure).

3.4.b. System state degradation

The second dimension, the *system state degradation*, is the difference between the achieved and expected system states after recovery. This dimension shows whether the user's goal has been fully achieved or not. It is an important issue from the user's point of view. We suppose that, at the second time, the user succeeds in performing the task without error .

3.4.c. Graphical editor example

Let us study a typical task in a graphical editor like MacDraw®. The user's task is to create a perfect circle. The user forgets to press the shift key before drawing, so the circle is in fact an oval.

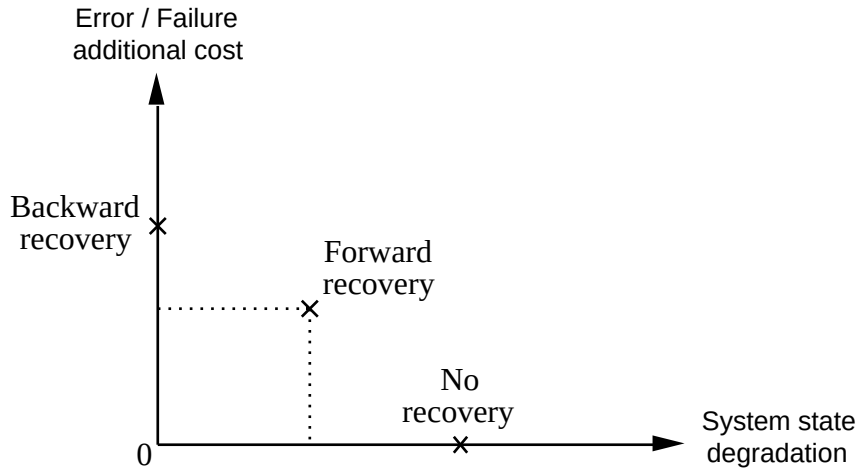

Figure 2: Error recovery in the graphical editor example.

The user can choose to manage the error by the undo function of MacDraw®, and re-draw the circle. This is a *backward error recovery*. As shown on Figure 2, the cost is high because the user must activate the undo function and to redraw. Moreover, the system state degradation is void because the goal, a perfect circle, has been achieved.

That's why the user can choose to turn his oval into an approximate circle by direct manipulation. This is a *forward error recovery*. As shown on Figure 2, the cost of the recovery task is low because the user has to perform a simple direct manipulation. Moreover, the system state is not optimal (the circle is not perfect) but enough satisfactory for the user's need.

The user can also choose not to recover the error. As shown on Figure 2, the cost of the recovery task equals zero, but the system state degradation is high because the circle is still an oval.

This example shows how a designer can evaluate, for every significant task, the possible recovery functions from the user's point of view according to our dimensions. Let us see their applications to time-critical systems.

3.4.d. Time-critical systems

In time-critical systems, the recovery functions are not always possible. The time is also an additional dimension of interest to think about recovery. In order to illustrate the importance of

time, we present the example of an aircraft pilot who has forgotten to extend the landing gear of an aircraft during the final path before landing.

As shown on Figure 3, the pilot can recover his error by backward error recovery – extend the gears – before T_1 and then land safely. The cost of this recovery is low and the system state is optimal: we deplore neither waste of time nor extra use of kerosene. Do note that the pilot can also choose to use a forward error recovery here, but it proves to be less interesting.

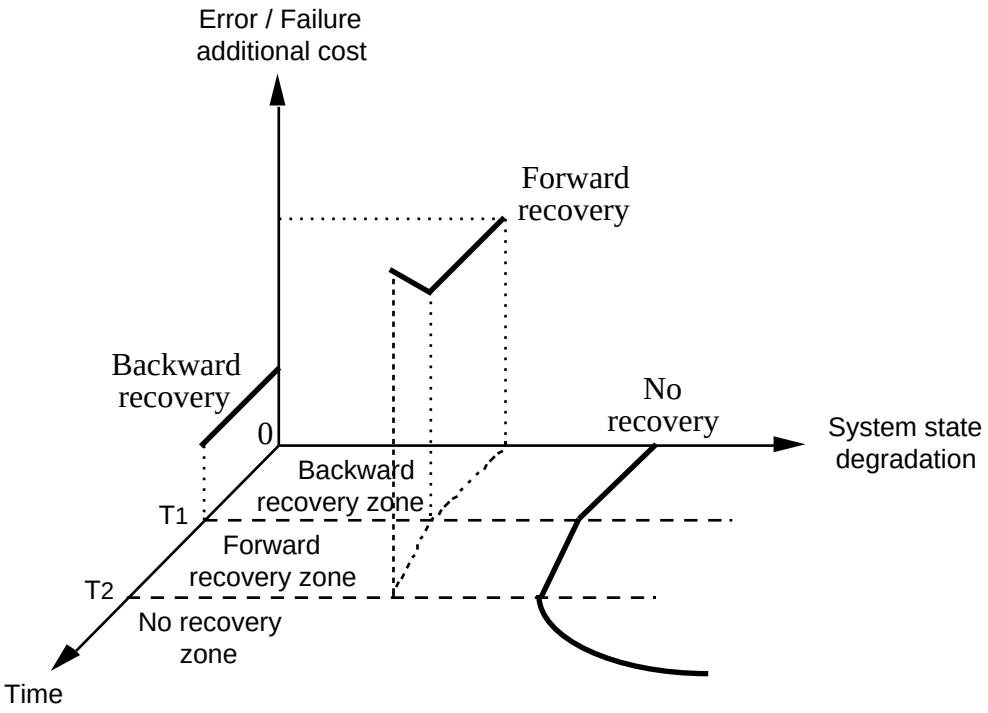

Figure 3: Error recovery in the landing aircraft example.

After T_1 the pilot does not have enough time to extend the gears before touching down the runway. So he has to go-around and to re-try the landing. This a forward error recovery: the cost is high for the pilot – the go-around task –, and the system state is not optimal because of waste of time and kerosene.

After T_2, it is too late, the aircraft is on the runway and the pilot has neither enough time to extend gears nor to go-around. The cost of the recovery equals zero but the system state degradation is high because the aircraft has crashed on the runway.

We assume that a graph can help designers visualizing the consequences of error recovery functions for critical tasks. These dimensions can also help designers to choose whether they implement or not recovery functions with respect to the cost the user will have to take care of.

CONCLUSION

As a conclusion, we hope we have provided designers a new way to understand the user's point of view in error recovery. We assume that these dimensions can also be used to understand system failure recovery in safety-critical systems. By doing so, we hope that these dimensions can ensure more safety in the conception of the interface of these systems. Of course, these dimensions can also be used in the design process of desktop computer interfaces in order to evaluate the real benefits of recovery. During the workshop, we would like to discuss the possible ways to include error management in the development process of user interface by the way of models and formalisms. Then, we would like to evaluate the usability of our dimensions of analysis in this context.

ACKNOWLEDGMENTS

The author would like to acknowledge the CLIPS-IMAG laboratory which partially support this work. Many thanks to Sylvie Ferrès for the English review of the article.

BIBLIOGRAPHY

Airbus Industrie. (1992). *Flight Crew Operating Manual A320*.

Dix, A., Finlay, J., Abowd, G. & Beale, R. (1993). *Human-computer interaction*. Cambridge, United Kingdom: Prentice Hall.

Dorwell, J. & Long, J. (1989). Towards a conception for an engineering discipline of human factors. *Ergonomics*, 11, 1513-1535.

Johnson, C. & Gray, P. (1996). Supporting Error-Driven Design, *Proceedings of the Third International Eurographics Workshop on Design, Specification, and Verification of Interactive Systems (DSV-IS'96)*, 207-228. Springer-Verlag.

Lenman, S. & Robert, J.-M. (1994a). A framework for error recovery, *Proceedings of the International Ergonomics Association (IEA'94)*, 374-376. International Ergonomics Association.

Lenman, S. & Robert, J.-M. (1994b). Investigating the granularity of the Undo function in human-computer interfaces. *Applied Psychology: An international review*, 4 (special issue on Human Errors), 543-564.

Leplat, J. (1985). *Erreur humaine, fiabilite humaine dans le travail*. Paris, France: Armand Colin.

Lewis, C. & Norman, D.A. (1986). Designing for Error. In D. A. Norman & S. W. Draper (Eds.), *User Centered System Design / New Perspectives on Human-Computer Interaction* (pp. 411-432). Hillsdale (NJ), USA: Lawrence Erlbaum Associates.

Ministère de l'équipement des transports et du tourisme. (1994). *Commission d'enquête sur l'accident survenu le 20 janvier 1992 à l'Airbus A320 F-GGED près du mont Sainte-Odile (Bas-Rhin)* (Rapport final ISSN 0242-6773). Journal officiel de la République française, édition des documents administratifs, n°31.

Nicolet, J.-L., Carnino, A. & Wanner, J.-C. (1990). *Catastrophes ? Non merci ! La prévention des risques technologiques et humains*. Paris, France: Éditions Masson.

Norman, D.A. (1990). *The design of every day things.* New York (NY), USA: Doubleday Currency.

Rasmussen, J. (1986). *Information processing and Human-Machine Interaction : An approach to cognitive engineering*. Amsterdam, The Netherlands: Elsevier Science.

Reason, J. (1990). *Human error*. New York (NY), USA: Cambridge University Press.

Swain, A.D. & Guttmann, H.E. (1983). *Handbook of human reliability analysis with emphasis on nuclear power plant applications* (Final report NUREG/CR-1278F). Nuclar Regulatory Commission (NUREG).

Yang, Y. (1992). Anatomy of the design of an undo support facility. *International Journal of Man-Machine Studies*, 1, 81-95.