

HUMAN-COMPUTER INTERACTION IN SAFETY CRITICAL SYSTEMS

BRIDGET A. LEATHLEY

Four Elements, Greencoat House, Francis St, London, SW1P 1DH, U.K.

SUMMARY

Increased dependence on computers in safety critical industries means increased dependence on a smaller number of people—both the users of computer systems, and those responsible for their design and production. Reliability and integrity of such systems can be enhanced without large overall increases in costs by looking critically at the human-computer interaction early in the design phase. In this paper, a selection of human factors principles are applied to safety critical system design. This includes making systems easier to learn to use, reducing alarm overload, matching the design of menus to the task, consistency and modality.

KEY WORDS: computer; safety; human factors; human-computer interaction; reliability

1. INTRODUCTION

With the advance of information technology capabilities, and increased cost of labour, the move to reduce the number of people involved in processes increases. While on the one hand this is believed to reduce the potential for human error, it also places greater dependence on a smaller number of people, including the people who produce the software, and those who make use of software-controlled systems. In addition, the consequences of one mistake by the remaining humans (again, either the software developers or the users) are increased. Hence, there is an ever increasing problem of dependence on software in safety critical systems, by people who do not fully understand software. We are very conscious of the need to improve and maintain integrity of systems, without driving up costs. This paper covers a number of issues concerned with the human users of computers. It draws on previously unpublished work in the area of human-computer interaction (HCI) design, evaluation and testing over a number of years, and guidelines for HCI which were set as long ago as the 1970s and 1980s.

2. LEARNING

2.1. *Simulators*

Training users to operate a safety-critical system correctly is more important than with non-safety systems. Much of the HCI literature encourages a 'learning by exploration' approach to software, e.g. training wheels,¹ and now ever increasingly multimedia supports the 'explore and play' approach. At first sight, this seems a dangerous way of encouraging users of safety critical software to improve their knowledge. Browsing in a multi-media encyclopedia

is a sensible way to learn what it can do, and how to use it; experimenting with the facilities in a word processor, and producing an incorrectly formatted set of letters the first time a user tries to use mail merge may waste some time, but it is reversible and hence is acceptable; halting production or worse by experimenting with parameters on a process control system is not. However, use of simulations in this way can be effective. The nuclear² and aeronautical³ industries have used simulators successfully for many years. The use of realistic simulators allows users to find out what happens when they try unauthorized moves, in safety, rather than experimenting with short-cuts on a live system (something nuclear industry users would not do, but which is known to occur in the process industries). Expense has deterred other industries from making use of simulators. However, even partial simulators can be valuable, and increasingly cheap and powerful computers can be used to simulate the functionality of operations, even if not every lever and button can be re-created.

2.2. *On-line help*

As well as training to improve knowledge, on-line information can encourage users to look up the answers to questions when they are not sure. Manuals left on shelves are less attractive to use than a well-designed, context sensitive on-line help system.⁴ Users are less likely to guess an answer if it is obtainable at the press of a button, than if they have to search through several manuals to find it. However, on-line help for any system must be well designed to be of use, and this applies even more so to safety critical systems, where the consequences of mistakes are even greater. The design of menu structures outlined in Section 5 can be applied to help systems as well as functional commands.

3. ALARMS!

Spurious alarms are one of the most demotivating aspects of many monitoring systems. Frequently, control room operators are surrounded by multiple alarm systems, added at different stages, and never designed to operate together. The operators learn to ignore red flashing lights, yellow rotating beacons, and blue noisy sirens, having established in the past that these are all within expectations. These lights and bells have been replaced in many control rooms with flashing icons on a computer screen. Again, operators learn that most alarms are spurious and of no concern. Similar problems of alarm overload have been noted in intensive care units and operating rooms in hospitals, where anaesthetists resort to turning alarms off on equipment, rather than being constantly distracted by them.⁵ Alert systems which constantly sound false alarms will demotivate attention, and inevitably lead to a delay in recognizing real emergencies when they occur. The application of existing guidelines on the number of, and discriminability of auditory alarms would go some way to reducing the problem,⁶ but in addition the overall design of systems has to take account of human capabilities. Workload studies are one way of achieving this.

4. OTHER TRADE-OFFS

The need to make trade-offs between different requirements is understood in the field of HCI. Although there are platform specific style-guides,⁷⁻⁹ the difficulty in producing definitive requirements for all user interfaces is that the solution depends on the nature of the application. However, knowing that a system is safety critical does lead the HCI practitioner to be able to make some recommendations as to appropriate trade-offs.

5. MENU DESIGN

The HCI community are divided about the relative merits of breadth versus depth^{10,11}. If there are (for example) 64 commands available to the user, these can be presented in a number of ways. For example, the user could be presented with a scrolling list of options. These could be sorted alphabetically, or in some other logical manner, and users scroll through until they find the required option. At the other extreme, the user can be presented with two options, which categorize all commands into two groups. Having selected one, two more, further sub-dividing the chosen group are presented, and so on until the final option is chosen. These extremes are illustrated in Figures 1 and 2. Numbers are used to indicate individual options on a menu. In between, there are options for two layers of eight items each (Figure 3), or three layers of four items (Figure 4). Irregular grouping of options may be preferable, in order to group commands logically (for example, with the

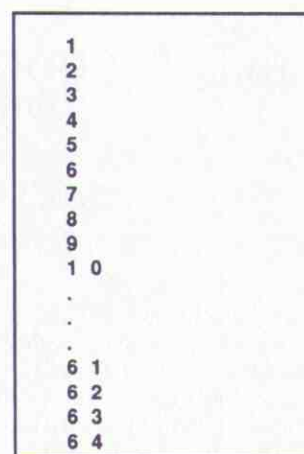


Figure 1. One layer menu

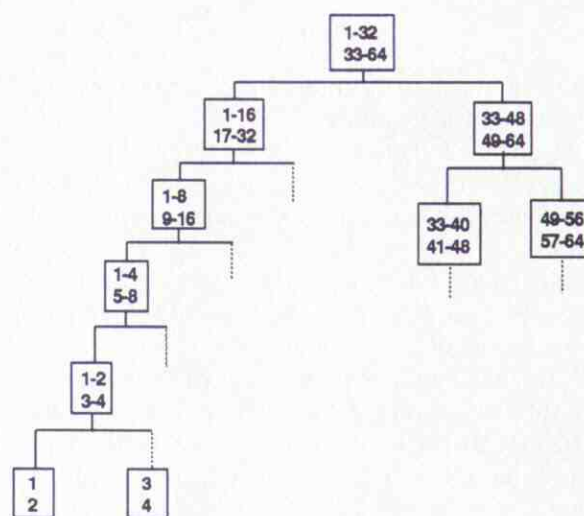


Figure 2. Multi-layer, two-option menu

three-layer menu, having some menus of three items, and others of 5).

One rule of thumb for menu structures,¹² is to have no more than 10 items on a given menu, and that the depth of tree should be three levels or fewer. Hence, the 64-item menu has too many items, and the two-item menu, too many layers. Two layers of eight items (Figure 3) and three layers of four items (Figure 4) both fit this model. However, when dealing with a safety critical system, this may not be the best menu design. For example, we can take advantage of the inaccessibility of the lower levels by placing commands which are not time-critical (i.e. are not needed quickly), but can have disastrous effects if used by mistake, at a fourth level. This discourages the user from using them inadvertently, and provides an extra layer of checking. As Figure 5 illustrates, this makes the tree uneven, something usually not favoured by HCI practitioners, as it may result in a user failing to use all the functionality of some word-processing system. In this case however, discouragement is exactly what is needed. A further rule to note is that it is advis-

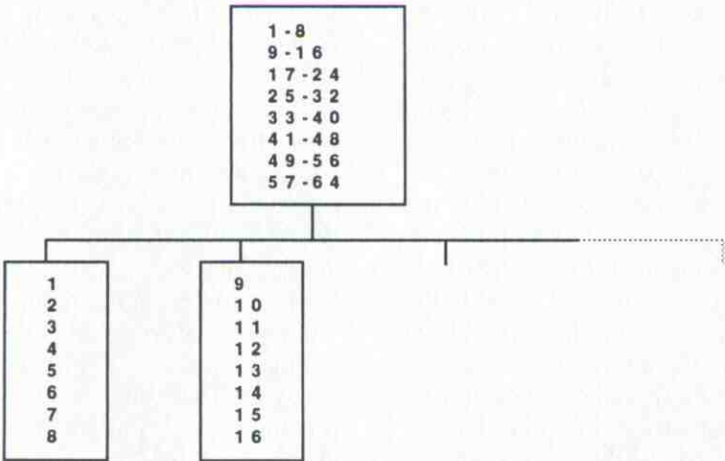


Figure 3. Two layer, eight-option menus

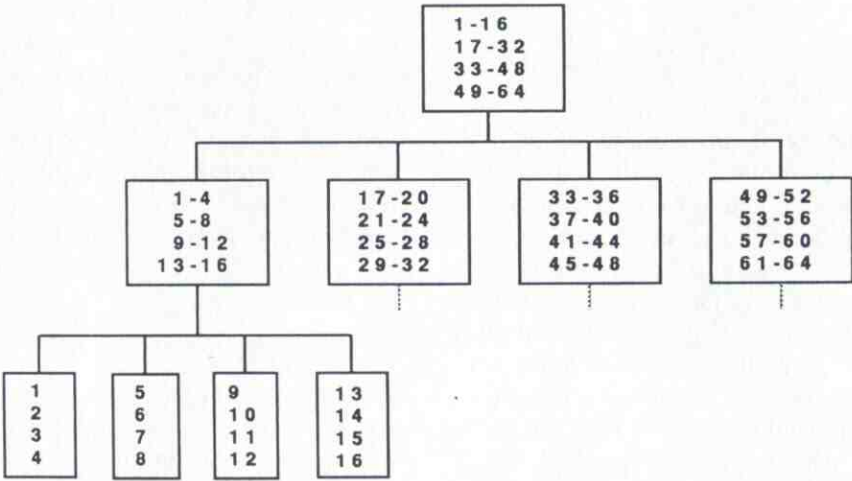


Figure 4. Three-layer, four-item menus

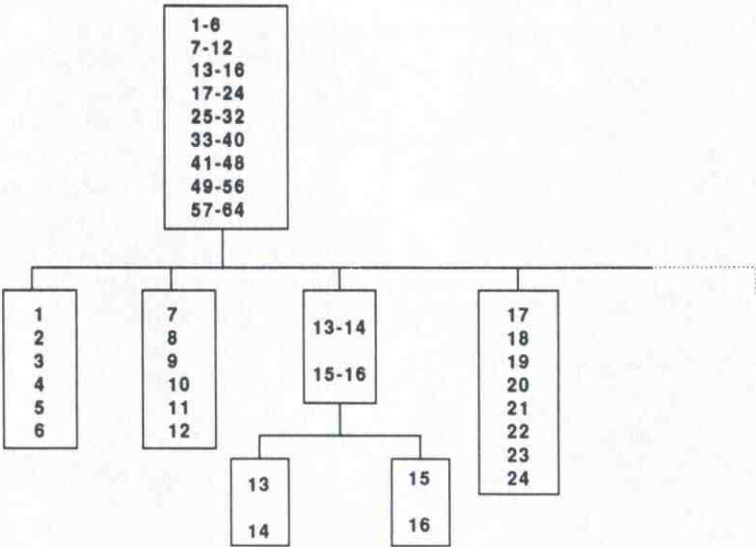


Figure 5. Uneven menu structure, with reduced accessibility of some options

able not to have both the title of another menu, and an option, on the same menu.¹³ For example, if the menu titles '13-14' and '15-16' were included

as part of the menu which also lists end options (e.g. 11, 12 etc) there could be some confusion over menu selection versus option selection.

Since the new tree is uneven, further advantage can be made of this, and commands which may be needed more quickly placed higher up the tree. For example, an option which displays important emergency instructions, or a command to stop a piece of moving equipment quickly following an accident, may need to be more accessible. If the top level menu offers an extra option of emergency commands, the top level menu still has fewer than the ten items allowed on a menu, but with the urgently needed commands now on the second level rather than the third (see Figure 6). Again, options which lead to another menu should not be confused with options which lead directly to an action. Hence, although placing 63 and 64 on the same menu as 49-52, 53-56, 57-59 and 60-62 would still result in a menu with fewer than 10 items, it might lead to confusion between an action and a further menu.

6. CONSISTENCY VERSUS REDUCED KEY STROKES

Another example to illustrate the principle of different trade-offs for safety-critical systems is the tension between consistency¹⁴ and reducing the number of key strokes.¹⁵ In some command-driven interfaces it is necessary to press the carriage return (or 'Enter') key in order to confirm a command. Where commands are of a variable length, some indication is needed to give the computer a signal of when to process the command. In other situations within the same system, the command may be of a fixed length. For example, if a yes/no question is asked, the user need only type in a 'y' or an 'n' to indicate a preference. Typing a carriage return as well improves consistency, but slows down use of the interface. Experienced users of such systems prefer some measure of inconsistency to improve speed of use. Novice users tend to prefer the consistency and

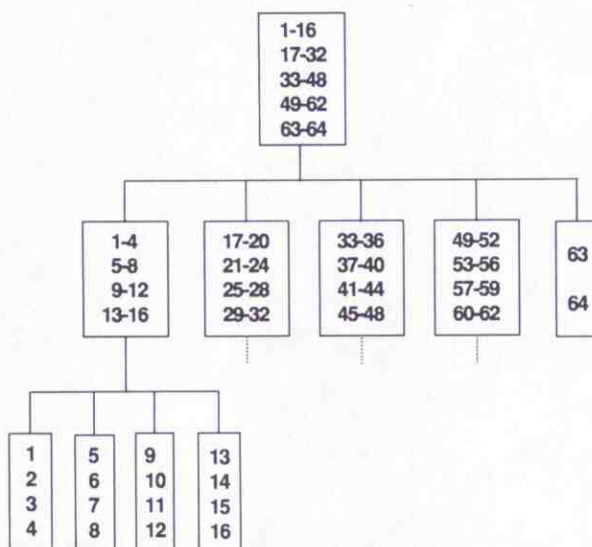


Figure 6. Uneven menu structure, with increased accessibility of some options

safeguard of being able to see the command and confirm it, at the cost of slower use. Hence, the solution for a safety-critical system, which we would assume would be used by experienced users, would seem to be to reduce key strokes, and sacrifice consistency. However, the consequences of inconsistency are likely to be greater, and the costs of slower use lower. In a typical non-safety critical application, where a user is entering lots of information, for example about customers, perhaps by talking to a customer over the phone, speed of entry is important. Use is intensive, and a few seconds saved on each entry adds up to processing more customers each day. A mistake is not critical. It may result in incorrect information occasionally, but the extra time taken to make occasional corrections is more than compensated for by the time saved on each entry.

However, in a safety-critical system, it is unlikely that success is related to the number of entries processed. The main concerns are checking and monitoring of systems, varying parameters accurately and so on. Forcing the user to delay before confirming an entry may actually produce benefits, since the adverse effect on safety and on productivity of one mistake is normally higher than the accumulated cost of several small delays in actions.

7. MODALITY

Another human factors recommendation is to avoid 'modes', that is, states that the system can be in which prohibit or permit different actions, even though looking superficially the same.¹⁶ As an example, consider a data entry form in a system for storing names and addresses, as illustrated in Figure 7. In order to look up a telephone number, the user perhaps selects a 'look-up' option from a menu, and then selects the required entry either by paging through, or by entering the name, and retrieving the record in that way. In order to create a new record, or to alter an existing one, the user may have to select a different option of 'edit entry'. Such a system would be modal if both the 'look-up' screen and the 'edit' screen looked like Figure 7, but in one the user could change the contents of the fields, but in the other they could not. A traditional way around this problem is to have only one function, which displays the entry as in Figure 7, and allows the user to change entries if they wish, or just to look at the information (or to make the call from the number, if the system is linked to a modem).

Taking a similar user interface problem in a safety

Name:

Address:

Phone number:

Figure 7. Data entry form for customer records

Pump id:			
Min temp:		Min pressure:	
Max temp:		Max pressure:	
Actual temp:		Actual pressure:	

Figure 8. Data entry form for process control

Pump id:			
Min temp:		Min pressure:	
Max temp:		Max pressure:	
Actual temp:		Actual pressure:	

Figure 9. Data entry form for process control, with edit control

critical application, Figure 8 presents a screen which both displays and allows the editing of some system parameters, thus avoiding breaking the 'no modality' rule. However, in such a system, accidental editing of a parameter could be disastrous. So is the solution to break the modal rule, and have two versions of Figure 8, one which can be edited, and one which cannot? As well as breaking the modality rule, it also introduces a further problem that the user may think they are in the safe 'read-only' screen, when they are actually in the 'change parameters' screen, resulting in accidental, and potentially catastrophic mistakes. Having a different title on the screen is not sufficient, as this is easily missed. Having a completely different layout for the two screens is not the solution either, since people rely on position of fields to identify them, and are more likely to make an incorrect reading.

One solution is to have the edit function available from the display screen. Figure 9 illustrates the display screen with an additional 'edit' button. Pressing the 'edit' button then allows the user to cursor into the field to be edited (the cursor should not move into fields which should not be changed). This could, of course, be achieved in other ways.

8. CONCLUSIONS

This paper has illustrated how existing guidelines and on-going research can help to provide solutions to just one area of safety critical software development, that of the design of the user-computer inter-

action. Involvement of HF personnel in projects, to assess the trade-offs in user interface design, to produce style guides for developers, and to improve the design of the user interface offers real benefits in terms of 'safer' software.

REFERENCES

1. J. M. Carroll, 'Minimalist design for active users' in H. J. Bullinger and B. Shackel (eds), *Human Computer Interaction—INTERACT '84*, Elsevier Science Publishers BV, North-Holland, 1985.
2. J. White, 'Simulation enters a new era', *Nuclear Engineering Journal*, September 1992.
3. T. Furniss, 'Team effort', *Flight International*, 16–22 February 1994.
4. G. Marchionini, 'Making the transition from print to electronic encyclopedias: adaption of mental models', *International Journal of Man Machine Studies*, **30**, 591–618 (1989).
5. K. Momtahan, R. Hetu and B. Tansley, 'Audibility and identification of auditory alarms in the operating room and intensive care unit', *Ergonomics*, **36**(10), 1159–1176 (1993).
6. M. S. Sanders and E. J. McCormick, *Human Factors in Engineering and Design*, 6th edn, McGraw-Hill, Montreal, 1987.
7. IBM, Systems Application Architecture, Common User Access, Basic Interface Design Guide (BIDG), 1989.
8. *OSF/Motif Style Guide, Revision 1.2*, Open Software Foundation, Prentice-Hall, 1993.
9. *Macintosh Human Interface Guidelines*, Apple Computer Inc, Addison-Wesley, 1992.
10. B. Shneiderman, *Software Psychology: Human Factors in Computer and Information Systems*, Winthrop Publishers Inc, Cambridge, Mass, 1980.
11. T. K. Landauer and D. W. Nachbar, 'Selection from alphabetic menu trees using a touch screen: breadth, depth and width', in L. Bormanand and B. Curtis (eds), *Human Factors in Computing Systems, Proceedings of CHI '85 Conference*, San Francisco, April 1985. ACM, New York.
12. Internal company standard.
13. S. L. Smith and J. N. Mosier, 'Guidelines for designing user-system interface software', *Technical Report ESD-TR-86-278*, The Mitre Corporation, Bedford, Mass, 1986.
14. Internal company standard.
15. Internal company standard.
16. Internal company standard.

Author's biography:

Bridget Leathley worked in the British Telecom Human Factors Division at Martlesham Heath, Ipswich, from 1986 to 1993, working on a wide range of telecoms related products, and managing the production of a company-wide User Interface Style Guide. In 1993 she moved to Four Elements, a Risk Management and Industrial Safety consultancy. Since then, she has provided Human Factors and Safety Management consultancy services to the railway, aeronautical, power, chemical and off-shore industries, as well as to departments of British and foreign governments.

Copyright of Quality & Reliability Engineering is the property of John Wiley & Sons, Inc. / Engineering and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.