NODDY'S GUIDE TO USABILITY

(to be published in Interfaces, the magazine of the British HCI Group)

Andrew Monk, University of York, UK

University of York, U.K., a.monk@psych.york.ac.uk

The progress that HCI has made in the last twenty years is simply amazing. HCI research has had an enormous influence on the software products that everyone takes for granted. For some reason, and I guess you have to blame the educators here, we often sell ourselves short. There is theory, there are methods and together they constitute a body of work that has changed the world for the better. We can engineer usability. Read on to see how.

WHAT IS HCI AND WHAT IS USABILITY?

Not everyone reading this article will know what HCI is. I should start at the beginning. Human-Computer Interaction (HCI) began as a discipline in the late 1970s and early 1980s. Initially it came about through an alliance between Computer Scientists and Psychologists. Since then Ethnography, Ergonomics and Activity Theory have all been recruited to the cause [14]. HCI research is concerned with how to ensure usability, that is to say, products that are effective, efficient and satisfying to use. HCI researchers try to understand what users want to do and how designers can be helped to provide products that satisfy these needs.

ISO 9241 AND VISUAL BASIC ARE THEORIES OF USABILITY

Table 1 lists the parts of the international standard ISO 9241. Parts 1 to 9 are broadly ergonomic but parts 10 to 17 are directly concerned with HCI design, how to ensure usability. An international standard has the weight of law behind it but perhaps a more commonly used form of standard is the "style guide". This rather misleading term is taken to mean a set of guidelines describing how a graphical user interface should work, for example, what a dialogue box should look like, how it should behave when the user interacts with it and when it should be used rather than some other device such as a menu. Apple produced the first style guide in 1987 [1, 2]. There are now style guides for all the commonly used graphical user interfaces (GUIs) including Microsoft Windows [11]. Style guides are supported by software tools. Thus a software developer using a programming tool such as Visual Basic will find it very much easier to obey the Microsoft Windows style guide than to ignore it. This prevents them from developing idiosyncratic interfaces that do not behave in the way users are used to. At the very least, by enforcing a degree of consistency in this way, style guides ensure that when a user learns to do something in one context that knowledge will transfer to new contexts in a sensible way.

Table 1. ISO 9241 Ergonomics requirements for office work with visual display terminals (VDTs)

Part 1 General Introduction
Part 2 Guidance on task requirements
Part 3 Visual display requirements
Part 4 Keyboard requirements
Part 5 Workstation layout and postural requirements
Part 6 Environmental requirements
Part 7 Display requirements with reflections
Part 8 Requirements for displayed colours
Part 9 Requirements for non-keyboard input devices
Part 10 Dialogue principles
Part 11 Guidance on usability specification and measures
Part 12 Presentation of information
Part 13 User guidance
Part 14 Menu dialogues
Part 15 Command dialogues
Part 16 Direct manipulation dialogues
Part 17 Form filling dialogues

So where did these standards and style guides come from? The answer is from years of painstaking HCI research. One of the first set of guidelines by Smith and Mosier [20] referenced all the papers that led to each of their 944 guidelines. As time went by authors concentrated on the guidelines and stopped providing the references but the research knowledge drawn on is there all the same. Style guides, and ultimately software tools, encapsulate a great deal of empirical and analytic work carried out by HCI researchers to find out what actually was the best way of doing things. In that sense they are theories of HCI. A software tool such as Visual Basic even meets the formal definition of a theory in that it constrains how something (a user interface) may look and behave. It constrains it in such a way that it is more effective, efficient and satisfying to use than it would have been if the design had not been constrained in this way.

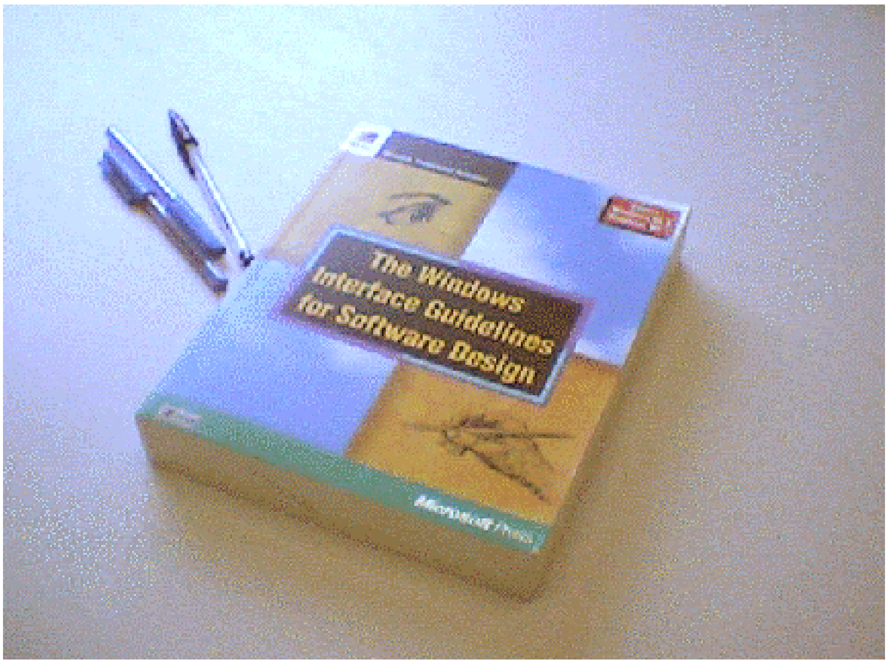


Figure 1. Is this a theory of HCI? I think it is, it's definitely fat enough!

PRINCIPLES OF HUMAN-COMPUTER INTERACTION

Early work on the effective use of graphical user interfaces was concerned with establishing higher level principles for good user interface design (see for example [10]). These principles are the basis of the more detailed style guides and are often re-iterated in them. Take for example the principle of "reversibility". One of the problems users had with early interactive systems was that they did not encourage exploration. Carroll and Carrithers [4] described how users might spend several minutes recovering from the wrong choice in a menu. To avoid this, style guides prescribe a variety of devices for undoing the unwanted effects of actions taken by a user, e.g.: the "back" button in a web browser; the "cancel" button in a dialogue box, or the "undo" function in a word processor. All these features follow the principle that the effect of any action that a user takes should be reversible. Users should be able to take this as given and where it is simply not possible the user should be warned before they take the action in the first place.

Another valuable principle that has been analysed in some depth is action-effect consistency (see my previous Noddy's Guide to Consistency, Issue 45, 2000; available from http://www-users.york.ac.uk/~am1/ftpable.html). This states that if the user takes some low level action it should have the same effect whatever the context. For example, pressing the delete key or clicking with the mouse should have the same effect whether one is editing a file name in a dialogue box or editing the text in a document. Another way of expressing this principle is to say that interfaces should be "mode free". In practice some degree of "modedness" is inevitable and the question is how to predict when modes will be a problem and how to signal them to the user [9].

Principles concerned with consistency in one form or another have been a recurring theme in HCI. "Task-action consistency" [17] is an attempt to optimise the relationship between a user's view of the task they are trying to complete, e.g., drawing a square, and the set of actions they need to take in order to complete that task. People expect tasks that they view as similar to require similar actions. Thus the actions required to draw a square must be consistent with the actions required to draw a circle.

Many of the problems people have with the new forms of interaction needed to work mobile devices such as cell phones can be readily understood, and fixed, by applying these principles and there is currently a renaissance in this research on design principles.

INTERNATIONALLY AGREED METHODS

Do you know how an international standard comes about? First a committee of experts, some of whom may be academics, writes down an agreed form of words - seems unlikely but they do. Then, and this is the staggering bit, they send this from of words to lots of other people, in different countries and with different vested interests, and these people "vote" on whether they agree with it too. If everyone does then the standard is published. Knowledge encapsulated in an international standard is mature knowledge. Everyone agrees it is right.

There is this level of general agreement on the processes needed to ensure effective user-centred design. The international standard ISO 13407 ("Human-centred design processes for interactive systems") specifies just what it says in the title. The same level of agreement can be seen in HCI text books [6, 18] and in published methodologies such as Contextual Design [3] and Monk's Light Weight Techniques [12, 15] (Do we allow this kind of blatant plug? - Ed.). These common elements are illustrated in Table 2 and the following paragraphs describe them in a bit more detail.

Many computer systems come to grief because they are not designed to perform the right functions and so it is important to get human factors input into the earliest stages of requirements analysis. The first two processes depicted in Table 2 are concerned with understanding the work context and the work to be supported. *Understanding the work context* involves identifying all the stakeholders and their concerns. Computer systems change the way people work, otherwise there would be no point in introducing them. It is thus possible to provide a system that supports one person's work very well while having side effects on the way work is done that make another person's work difficult or even impossible. Only by identifying all the people that could be affected by the introduction of the new system and their particular concerns, is it possible to avoid this kind of problem.

Table 2. Common processes in user centred design

Understanding the work context

Methods: focus groups, interviews, observation

Representations: the rich picture

Understanding the work

Methods: focus groups, interviews, observation

Representations: HTA, WOD and exceptions, scenarios

Testing a top level design against your understanding of the work

Methods: Scenario walkthrough, Cognitive Walk Through

Representations: Story boards, dialogue modelling

User testing of more detailed prototypes

Methods: Usability Labs., Cooperative Evaluation

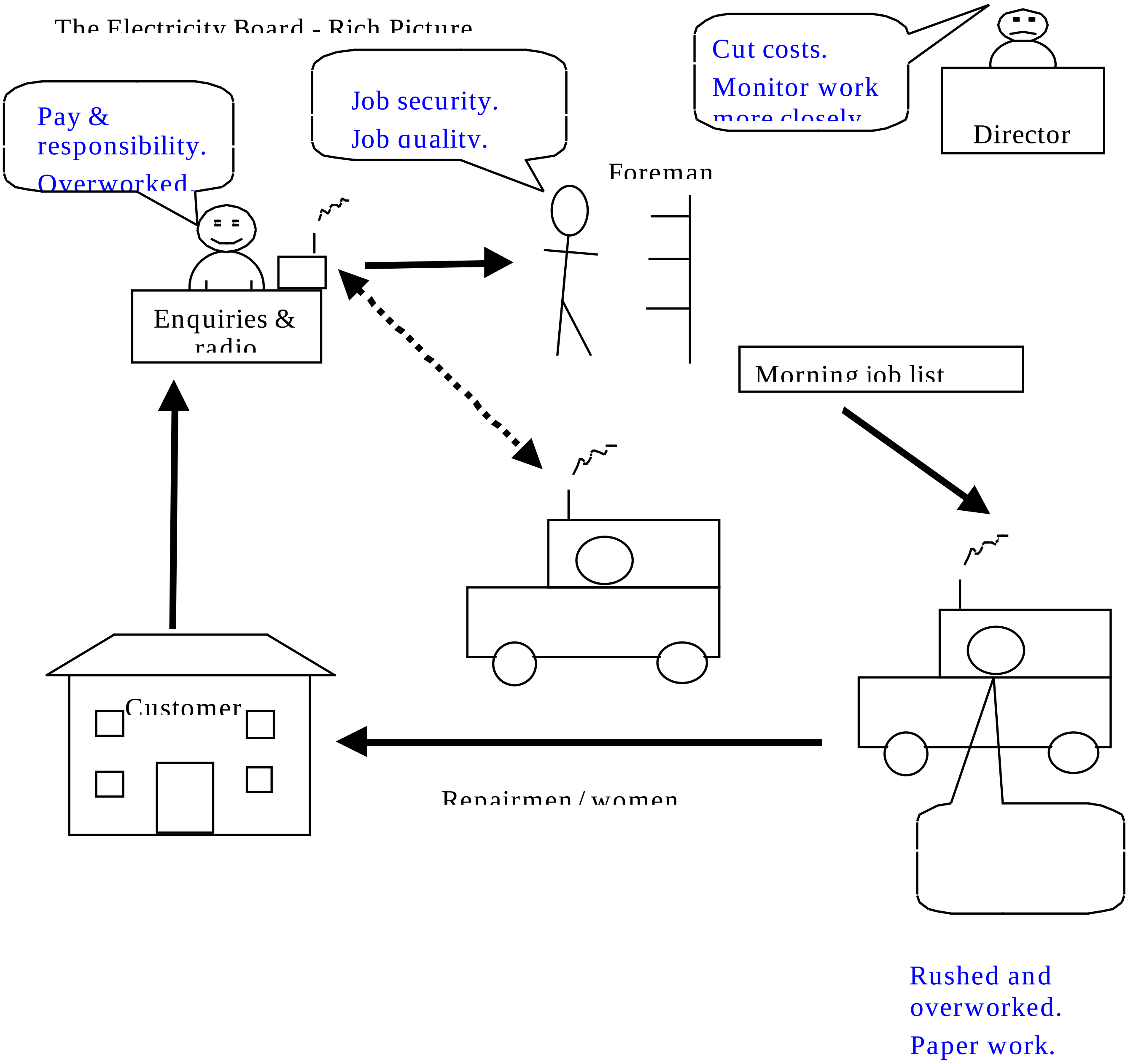
Representations: Paper prototypes, simulations

Understanding the work. Once the design team has gained a broad picture of the work context they can focus on the particular work to be supported by the computer system. As with the work context, the data used to do this will come from interviews and observation in the work place. Typically some sort of representation will be used to record and reason about the way the work proceeds. The two most commonly used are Hierarchical Task Analysis [19] and scenarios [5]. A scenario is simply a story that takes the reader through the steps taken to perform a work task described at a fairly high level. It should include details obtained from the analysis of the work context such as interruptions and parallel tasks not to be supported by the computer. In general several scenarios will be needed to cover the most important variations in the way work may be completed.

Testing a top level design against your understanding of the work. The next step is to build a model of the high level structure of the user interface. This will omit many details of screen design but will describe how a user moves from one task to another. This "dialogue model" [12] can be evaluated against the representation of the work to be supported. For example, one can go through the scenarios checking that all the work tasks can be completed and that the way the operator has to work is efficient and fits in with the larger job.

User testing of a more detailed prototype. Finally, a detailed prototype of the user interface is built and tested with real users. Much can be done at early stages using mock-ups or paper prototypes before any code has been written [15]. There are also usability inspection techniques that can be applied to a user interface specification [16]. In this way one can ensure that the user interface will communicate the designer's intention to the user effectively.

Figure 2. The Rich Picture lists the major stakeholders, their concerns (in speech bubbles) and a wide angle view of the work. This is one of the notations that can help designers reason about a design.



Different authors describe these four processes in different ways, and some add bows and frills of various kinds. However, they all agree on the basic steps, what they are to achieve and the order they should be carried out in. The disappointing thing is that not everyone out there uses them. Perhaps the real challenge for HCI is convincing people that we know what to do and that it is worthwhile to do it.

THE FUTURE: BROADENING THE CONCEPT OF USABILITY

The HCI knowledge I have described is old stuff and applies mainly to graphical user interfaces for office systems. Mobile and ubiquitous technologies are taking the computer out of the office into the street and into the home. Suddenly the landscape is unfamiliar. It took ten years to get from the first papers describing the problem of designing interactive systems for the work place (see for example [7]) to the first papers describing key concepts and methods (see for example [8]). It took a further 10 years for the area to mature to the extent there was sufficient consensus for clear standards to emerge.

It is to be hoped that our understanding of this new stuff will take less than 20 years. It is no longer hard to convince the people that matter that HCI issues are crucial to the success of their product. Also some of the old stuff will still be useful. Our research at York is to broaden the old conception of usability as "ease of use", "ease-of-learning" and "task fit". For example, many of the things we do in the home have no underlying task goal, we just do them for the experience they provide [13]. Neither is there the same level of agreement and encapsulation of the large body of research knowledge that exists on how we should use technology for communication and co-operation. Lots to do then, there is another world out there for us to change.

Acknowledgements

My thanks to members of the York HCI Group, particularly Michael Harrison and Peter Wright for useful comments and discussion while preparing previous versions of this paper.

REFERENCES

1. Apple *Human Interface Guidelines: the Apple Desktop Interface*. Addison-Wesley: Reading, MA, 1987.

2. Apple *Macintosh Human Interface Guidelines.* Addison-Wesley: New York, 1993.

3. Beyer, H. and Holtzblatt, K. *Contextual design: defining customer-centered sysytems.* Morgan Kaufman: San Francisco, 1998.

4. Carroll, J.M. and Carrithers, C. Training wheels in a user interface. *Communications of the ACM*, 27, (1984), pp. 296-308.

5. Carroll, J.M. and Rosson, M.B. Getting around the task-artifact cycle: how to make claims and design by scenario. *ACM Transactions on Information Systems*, 10, 2 (1992), pp. 181-212.

6. Dix, A., Finlay, J., Abowd, G. and Beale, R. *Human-Computer Interaction*. Prentice Hall: Hemel Hempstead, 1998.

7. Gaines, B.R. and Facey, P. Some experience in interactive system development and application. *Proceedings of the IEEE*, 63, (1975), pp. 894-911.

8. Gould, J.D. and Lewis, C. Designing for usability: key principles and what designers think. *Communications of the ACM*, 28, (1985), pp. 300-311.

9. Harrison, M.D. and Dix, A. A state model of direct manipulation in interactive systems. In *Formal methods in human-computer interaction,* Harrison, M.D. and Thimbleby, H., (Eds.), Cambridge University Press: Cambridge, UK, 1990.

10. Harrison, M.D. and Thimbleby, H.W. Formalising guidelines for the design of interactive systems. In *People and computers: designing the Interface,* Johnson, P. and Cook, S., Ed., Cambridge University Press: Cambridge, UK, 1985, pp. 161-171.

11. Microsoft *The Windows interface guidelines for software design.* Microsoft Press: Redmond, 1995.

12. Monk, A.F. Lightweight techniques to encourage innovative user interface design. In *User interface design: bridging the gap between user requirements and design,* Wood, L., Ed., CRC Press: Boca Raton, 1998, pp. 109-129.

13. Monk, A.F. User-centred design: the home use challenge. In *Home informatics and telematics: information technology and society,* Sloane, A.and van Rijn, F.., Eds., Kluwer Academic Publishers: Boston, 2000, pp. 181-190.

14. Monk, A.F. and Gilbert, N. *Perspectives on HCI: diverse approaches.* Academic Press: London, 1995.

15. Monk, A.F., Wright, P., Haber, J. and Davenport, L. *Improving your human-computer interface: a practical technique*. Hemel Hempstead: Prentice-Hall, BCS Practitioner Series, 1993.

16. Nielsen, J. *Usability engineering*. New York: Academic Press, 1993.

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

18. Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. and Carey, T. *Human-Computer Interaction*. Addison-Wesley: Reading, MA, 1994.

19. Shepherd, A. Task analysis as a framework for examining HCI tasks. In *Perspectives on HCI: Diverse approaches,* Monk, A. and Gilbert, N., Ed., Academic Press: London, 1995, pp. 145-174.

20. Smith, S.L. and Mosier, J.N. *Guidelines for designing user interface software*. Mitre Corporation: Bedford, MA, 1986.