Methods For Evaluating User Interfaces

Andy Holyer
School of Cognitive and Computing Sciences,
University of Sussex at Brighton
Falmer,
Brighton
BN1 9QH
UK
eMail: andyh@cogs.susx.ac.uk

November 10, 1993
Cognitive Science Research Paper No. 301

1.0 Abstract

There are a number of problems with the current state of the art in the evaluation of user interfaces. Firstly, there have been proposed a wide range of different methods for performing the evaluation, each with its own body of advocates, but with no common agreement as to how the different methods compare with one another. Secondly, the methods which exist at the moment either provide insufficient information or they are so time-consuming as to not be practical in real cases. Finally, many methods are unreliable, with individual evaluators having an undue influence on results, and there is in general no way to translate results into solid recommendations for improving the interface under test. This paper surveys a representative subset of evaluation methods, and suggests ways in which their differences may be resolved.

2.0 Introduction

It is almost universally accepted in Human-computer Interaction (HCI) that a reliable and efficient method to evaluate user interfaces is crucial to the design of better interfaces. However, there is almost equally universal disagreement as to how one should go about this: which of a number of competing procedures is the 'correct' one, and even where in the design process is the best place for evaluation to take place. This paper describes a selection of the better-known methods in an attempt to allow them to be compared on a fair basis. In writing the earlier sections of this paper I have, as far as possible, tried to keep away from the disagreements between different philosophies, since I find it unhelpful to any process of comparison. However, if in a few places my own prejudices show through, I have tried to keep these lapses as rare as possible.

3.0 Ways of categorizing Evaluation

ray, 1987]; [Macleod,1992]. The documents cited characterise differing methods in functional terms; that is, they categorize methods with respect to the sort of processes they involve, whereas I take a different tack.

Human-Computer Interaction is a new discipline, and there are (as yet) very few "native" HCI workers - that is to say, workers for whom HCI was their original subject of study. In recent years, it is true, graduates have begun to appear whose degree title is "Human-Computer Interaction" (or something similar), but all of these degrees are specialised courses within a department of Computer Science, say, or a department of Psychology. The vast majority of HCI researchers are Computer Scientists, or Psychologists, or Sociologists, or Linguists, who have since moved into HCI, and as such they enter the field with widely differing backgrounds in terms of investigative procedures and the like. When they come to examine the problem of evaluation, it appears that the procedures with which they are familiar play a significant role in their approach to evaluation.

I choose to divide the field of Evaluation in terms of the discipline from which each method is derived. In practice, this categorization is not completely clear-cut - for example, the method of evaluation by means of usability questionnaires is practised by experts in a number of fields - but I find it a useful tool for viewing the whole field.

4.0 Cognitive Psychological Approaches

The discipline of cognitive psychology has contributed two broad families of evaluation methods, which share similar underlying principles; one family, 'GOMS' (Goals Operators Methods and Selection Rules) is mainly practised in the USA; the other, 'Task Analysis' (TA), is commonest in the UK. The original description of GOMS is to be found in [Card, Moran & Newell, 1983]; the most comprehensive reference on TA methods may be found in [Diaper, 1989].

Both TA and GOMS work on the assumption that a user's interaction with a system consists of a set of high-level 'goals'. For example, in an electronic mail system, the user could be considered to have potential goals such as 'Read my mail', 'Organise old messages in an understandable fashion', or 'Reply to mail messages'. These goals may be decomposed into what GOMS refers as 'operators', that is, individual subcomponents which are independent of one another, which in combination will satisfy the goal. These operators may themselves be decomposed until eventually we are left with a set of indivisible 'methods'. In many cases two or more methods will satisfy a high-level goal: in this case a 'selection rule' comes into force to determine which potential method is actually applied.

For example, when editing a mail message the user wishes to delete a couple of paragraphs of text. There are at this point a number of potential ways to do this: (a) Move to the start of the text to be removed, using cursor keys, then use the control key for 'delete paragraph' until text is deleted; (b) Stripe over the desired text with the mouse, hit 'delete'; (c) Stripe the text with the mouse,

select ‘cut’ from the pull-down menu; (d) Move (by some means) to the end of the text to be deleted, and hold the delete key down; and so on. Note that not all of these methods will actually be available in any single application, and that it is not uncommon for overlapping subsets of these to be available on applications used on a single workstation. Given the above set of methods for solving the goal, GOMS theory proposes that some sort of ‘selection rule’, like that found in a production system, comes into force in the human cognitive system to decide which method is actually executed.

An evaluation using TA or GOMS usually starts from either a transcript of an actual interaction or a more-or-less formal description of the way a system under test operates.[1] This description is then manipulated so that instead of a ‘flat’ linear description of events, the evaluator arrives at a richly-structured description of the cognitive processes going on during the interaction. It is at this point that the various competing TA / GOMS formalisms diverge in their approach to the evaluation.

For example, the so-called ‘Critical Path Method-GOMS’ (CPM-GOMS) model [Gray et al., 1992], takes an analysis, applies time constraints derived either experimentally or by derivation in accord with Card Moran & Newell’s ‘Model Human Processor’ (see [Card, Moran & Newell, 1983]), and then submits the resulting network to a critical path analysis. In this way, it is possible to determine areas in a system which are especially ‘critical’ (for example occasions when system delays cause the user to wait for the system to ‘catch up’, or

semantic consistency, and in terms of the degree of alignment between syntactic and semantic structures. TAG itself is based on the work of Thomas Moran, in particular his Command Language Grammar [Moran, 1981].

There exists great contention within the HCI community as to the value (or otherwise) of TA and GOMS methods - indeed the entire discipline sometimes appears polarised between "TA supporters" and everyone else. The single largest problem faced by the supporters of Task Analysis methods is that the analysis process in extremely time-consuming. Benyon [Benyon, 1992, i] comments, " *as ana s s s expensive n t ean e ort- It s open a tte t at t s sort o ana s s s on w n e an requ r s ons ra etra n n an experts e D a per D aper , e onstrates t s w en us n AKD- epa s o text an repre sentat on w ar e requ r e n s s a exa p e, pr ar eause t e t o s otto up annot e ust e n t e eve o ana s s w epro u e - In ee D a per a ts t at t esa er su t ou ave een s ove e u or equ -*

5.0 Social Psychology Methods - Interviews and Questionnaires

Many HCI workers with an interest in evaluation come from a background in social psychology: hence it is natural that they should be inclined to use those methods which are most familiar to them in the process. This may go some way to explain the great popularity of questionnaires, and in structured interview techniques based on similar principles, in UI evaluation. In any case, there have been published a wide range of questionnaires designed to solicit users' responses to user interfaces.

A typical example of the more complex type of questionnaire is University College Cork's SUMI, the "Software Usability Measurement Inventory", which was produced as part of the ESPRIT project "MUSiC". SUMI is a 50-item questionnaire which can be scored in one of two ways: by means of a marking stencil supplied in the basic kit (SUMI is physically extremely well-produced: unlike many "evaluation tools", purchasers receive a very professional-looking kit, all packed in a pretty blue-and-white box), or by means of a software package which runs on an MS-DOS PC.

SUMI provides an overall Usability score, along which 5 sub-scales derived using factor analysis methods: "Affect" (a measure of how much the users found working with the system to be "pleasurable"), "Efficiency" (to what degree that considered their use of the system to be "productive"), "Helpfulness" (how much help they perceived that the system gave them), "Control" (their subjective feeling of "being in control"), and "Learnability" (how easy they felt the system was to learn). In addition, the professional version comes with a great deal of additional data which allows the PC software to indicate in which areas the interface under test scores above or below particular "expected" scores for typical software packages; thus a team working on (say) a Word Processor could find out whether their prototype scored better, worse or around average when compared with the average scores of other word-processors on the market.

Since it is the result of an ESPRIT project, SUMI has another advantage which in certain cases may be important: it is available in most of the common EC languages. This allows subjects to answer a questionnaire in their own language, which may in some cases greatly increase the ease of administering the questionnaire. SUMI is quite expensive financially, compared with many of its competitors: U.C. cork charges from 150 punts for a pack of 50 questionnaires, scoring stencils etc., up to 1000 punts for the full electronic version. For more details of the SUMI, see [Kirakowski et al., 1992].

At the other end of the scale, there is the 'System Usability Scale' (See Appendix A: on page 16.), which is the work of John Brooke of DEC. In 1985, as part of Digital's Integrated Office Systems programme, Brooke needed a "quick-and-dirty" way to assess the overall user response to a particular interface. He assembled a list of 50 potential questions graded on a Likert scale into a very long questionnaire, and tested this protoquestionnaire on 30 users with respect to two DEC. products, "chosen to be at extremes of usability for a given technology". Finally he selected those questions which correlated most strongly with a 'good' or 'bad' result. He chose the best ten, five positive, five negative, and Constructed SUS from alternate positive and negative questions.

The great advantage of the SUS is its astonishing ease of use. Fitting easily on a single side of paper, it is quick and easy to administer, and since it is less intimidating than longer questionnaires a far higher percentage of users are likely to actually complete it - a serious issue when you consider that many more "serious" questionnaires constitute a substantial document in their own right. It is also extremely quick and easy to score: simply rate each response on a scale of 1 to 5 (positive questions give a 5 for "strongly agree"; negative questions a 5 for "strongly disagree"), and average out the results. The SUS can be scored by hand in around 30 seconds. The surprising thing is how good the results are for such a simple process: the SUS has been examined by U.C. Cork and its results have been found to correlate with those of SUMI with a reliability of 0.8588, which considering that it is far simpler and cheaper to administer is very impressive.

In marked contrast to both of the above, there is a school of evaluation which exports the "Interface Checklist" approach from the field of Engineering Methods (q.v.), and attempts to customise the approach for a novice audience. A notable example of this is [Ravden & Johnson,1989], which is a quite substantial book essentially consisting of a single extremely long-winded questionnaire.

6.0 Social Science Methods

A number of methods of data-gathering derived from the social sciences have been applied to the problem of evaluating computer systems. The most active group in this respect appears to be that headed by Steve Draper in the Department of Psychology at the University of Glasgow. They have produced a large number of publications, of which the most general is [Draper, Gray,

Kilgour & Oatley, 1991]. A number of the more common of these methods is described below.

6.1 Talk-Aloud Protocol

Talk-Aloud is one of the most popular observational evaluation methods. It

cally by for example a one-way mirror or a video link. In any case it is often a good idea to video the session for later review.

The group leader demonstrates the system under test to the subjects. Note that it is not necessary for the system to be fully working for it to be demonstrated: a partially-completed prototype will often produce useful results, as will an explanatory video or even a paper-based mock-up. The group leader then leads a semi-structured discussion of the subjects' feelings about the system under test.

One great advantage of focus groups over similar methods is that the subjects outnumber the investigator(s), and hence the noted tendency to alter their testimony in order to 'appease' the investigator is greatly reduced. Focus groups also have the advantage that with a comparatively small amount of effort it is possible to get a large amount of useful information, and, most notably, that focus groups often point out faults which the investigators had not even thought of.

The largest single problem with focus groups is that they are highly dependent on the interpersonal skills of the principal investigator - it is very important that the investigator allows the subjects sufficient freedom to genuinely express their views, while at the same time ensuring that the discussion is sufficiently directed that the points the investigation team are interested are covered, and at the same time that no single subject is allowed to dominate the process - a very common occurrence.

6.3 Cooperative Evaluation

Cooperative evaluation is a process derived from Talk-Aloud, but in which instead of the volunteer being a passive "subject" to be observed by the evaluator, there exists an active dialogue between the evaluator and the volunteer as to how the interface could be improved. To quote [Wright & Monk, 1991]:

> *In evelop n t et n a ou e o t ea was to a et e pro e ur eas natura as poss e eus er s to to t n o or er se not as an exper enta su et ut as a o eva uator o t eputat ve s ste - en pu e t eus er s e av our t e eva uator a as que st ons su as at w t es ste o at as t one as t on et at at are ou tr n to o ou o t at ar t eus er a as t e eva uator que st ons t ou t e r ep es s ou e es ne to n out a out t eus er s pro e rat er t an to answ er t e que st on r et -*

As Wright and Monk themselves acknowledge, in principle this does not differ very much from conventional talk-aloud in practice: as they say, describing an evaluation exercise using talk-aloud, "Examination of audio tapes of the evaluation sessions reveals that at least some of the evaluators who were supposed to be using [...talk-aloud...] were interacting quite freely with the users".

Cooperative Evaluation appears to have been developed largely independently of similar work on social methods. In its favour, it presents a coherent and readable set of instructions for a would-be evaluator to use. Monk & Wright's definitive book on cooperative evaluation is expected at the moment.

7.0 Engineering Approaches

This group of evaluation methods borrow heavily from practices in conventional Software Engineering - heuristic evaluation from the use of checklists, and cognitive walk-throughs from the sort of paper walk-through familiar to most programmers. In [Macleod,1992] they are grouped together under the generic classification 'Expert Methods', since they all require a certain amount of experience on the part of the evaluator or evaluators in the problems likely to arise with user interfaces. This feature is, in my opinion, the most serious fault that these methods collectively suffer from: for economic reasons, it is rarely practical to keep such highly-trained (and hence expensive) staff on hand, when they do not appear (at least at first sight) to play any direct role in the design process. This is, admittedly, a problem faced by evaluation as a whole, but because of the very high degree to which the utility of this type of method is dependent on the skills of the evaluators, I would suggest that it is more serious is this case than in certain more empirical approaches, where at least some of the more time-consuming parts of the evaluation process may be parcelled out to assistants.

7.1 Heuristic Evaluation

The term 'Heuristic Evaluation' is used primarily by Jakob Nielsen to describe an evaluation method which initially appears so obvious that one might not consider it an 'Evaluation Method' at all: a 'User Interface Expert' examines the interface under test (or a more-or-less exact specification of it) and, from his or her experience, possibly aided by guideline documents such as [Apple, 1987], suggests potential problems which may arise.

There is evidence that this approach to evaluation is at least as effective as more elaborate methods, and is almost invariably a great deal faster. [Jeffries et al., 1991] compared four different evaluation methods and found that *Heur s t Eva uat on , oun t e ost serous pro e s w t t e east a ount o e ort'* **. However [Jeffries & Desurvire, 1992] indicates that this effectiveness is strongly dependent on the skill of the individual evaluator, and that in addition this effectiveness is an aggregate of multiple evaluations, rather than being the result of a single evaluation. They also noted that** *Heur st eva ua t on sse a o t epro e s oun n t e a orator . an usa t test n sse a out t esa enu er o pro e s oun n eur st eva uat on* **.**

This implies that, whilst undoubtedly being a useful tool for designers, heuristic evaluation is not the panacea which some have suggested it to be.

7.2 Cognitive Walk-throughs

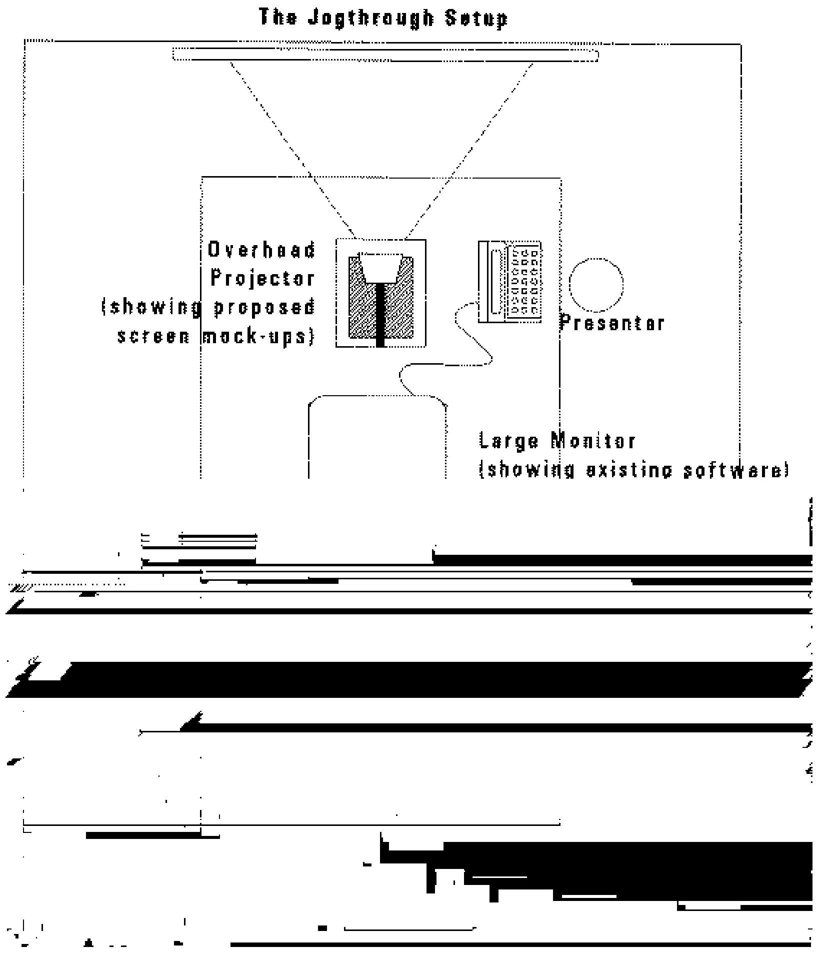
The Jogthrough Setup
Overhead Projector (showing proposed screen mock-ups)
Presenter
Large Monitor (showing existing software)

FIGURE 2. The Setup for a "Cognitive Walk-through". Reproduced from [Rowley & Rhoads, 1992]

The Cognitive Walk-through is a methodology developed to attempt to transfer academic theories of interface use into a form usable in an industrial setting. It models itself on the traditional software engineering practice of finding bugs in a program by performing a 'paper run' - stepping through the code stage by stage while keeping track of the flow of control, the value of variables etc., on paper.

The cognitive walk-through has evolved into a vigorous and formal process, which has been found to be very slow and expensive in terms of manpower for all but trivially simple interfaces. It consists of a number of skilled evaluators who examine a prototype interface (normally working with either a 'working' mock-up or with the actual interface). The evaluators step through the interface, at each point examining all possible actions which a user may make at each point in terms of a number of formal criteria. These criteria are usually fixed through filling in a series of forms for each state of the interface (for an example of the sort of form involved, see Figure 3 on page 10). By working through the possible states of the interface in this way the evaluation team can highlight elements of the interface which should be improved.

As I observed above, the process of performing a Cognitive Walk-through is a very long labour-intensive process. [Wharton et.al, 1992] comment that evaluators, *found the repetitive form fill-in to be very tedious enough so that it discouraged some evaluators from using the method in the future*. They also comment that *the walkthrough methodology presupposes more knowledge than most software developers have* — hence it is frequently necessary to have specialised staff simply to perform Walk-throughs (at extra cost).

There are also underlying assumptions in the walk-through methodology which limits the range of possible interfaces to which it is appropriate.The method was designed for use with ‘occasional use’ interfaces such as ATMs (Automatic Teller Machines — high street cash dispensers), where the length of any individual interaction is typically quite short, and where there is a limited set of paths that the interaction may follow. Attempting to apply this sort of method to a general purpose interface such as a word processor, where there may be hundreds of possible actions at any particular time, and where the interaction may continue for hours at a time would thus be virtually impossible.

In addition there is the tacit assumption in this method that the interface under test can be conveniently divided into discrete ‘states’, with discrete transitions between them. While in many cases this assumption is true, in others it is virtually impossible — for example in the case of trying to examine a direct

programmer as being a fellow designer / programmer. Due to the scale of investment required by many current evaluation methods, and also due to the somewhat different mix of skills required for evaluation from those required for programming, it is very common for evaluators to be cloistered away in Usability Labs or Human Factors departments, and then wheeled in on a project when required. This is a perfect breeding ground for the 'kill the messenger' effect, and it isn't helped by the fact that in industry human factors specialists often seem to be female while the design teams are usually predominately male.

When it comes to doing something useful with the results of an evaluation, there appears to be some sort of trade-off between the ease of administration of a method and the amount of detail produced by it. For example, question-

geous, assuming resources allow, to combine a number of different evaluations of an interface to ensure broader coverage. A heuristic evaluation of a prototype, for example could be combined with the use of focus groups or talk-aloud to find potential trouble-spots, which could then be examined in detail by means of more formal procedures such as task analysis or cognitive walk-through. At the same time, a mailshot of questionnaires amongst the alpha-testers (you do have alpha-testers, don't you?) will ensure that the evaluation process is at least on the right lines.

9.0 Prospects for Future Research

At the moment a lot of speculative work in Evaluation is aimed towards speeding up existing evaluation processes by partially automating them. For example, work at the National Physical Laboratory under the MUSiC project has produced a package called DRUM which speeds up the process of producing an action log from a video-tape of an interaction (See [Macleod and Rengger, 1993], and also Figure 4 below). DRUM is a (very sophisticated)

FIGURE 4. The Video Analysis tool "DRUM": The Recording Logger. Reproduced from [Macleod and Rengger, 1993]

Hypercard stack which controls a video recorder, allowing an evaluator to move back and forward through a recording of a session frame by frame, adding annotations. It is claimed to speed up the process of analysing video data by a factor of 4 or 5 in the hands of an experienced user.

A second approach to solving the data bottleneck is to use intelligent tools to 'filter' the recorded data in order to isolate the interesting sections from the rest. Some work on this subject has gone on in recent years at the University of York (see [Finlay & Harrison, 1990]), where they have used pattern matching techniques (in this case an associative memory system known as ‘ADAM’ to extract ‘critical incidents’ from a log of user actions.

Another approach to the problem of keeping up with the flow of events consists of redefining the terms in which events are observed. Draper’s group at Glasgow is working on an alternate system of coding for an interaction which has been optimised in terms of Direct-Manipulation interaction (one problem with many current methods which make use of some form of event logging is that they were originally developed to analyse character-based interfaces, and in many cases can appear somewhat clumsy when faced with a WIMP interface). Draper has developed a logging tool with which he claims it is possible to log an interface in real time.

In my own research on User Interface evaluation, I have found myself more and more reminded of an anecdote told by Brenda Laurel in her book “Computers as Theatre”. She tells of a conference on highly interactive interfaces which she attended in which virtually every session was marred by a long and unproductive debate on what exactly was meant by ‘interactive’. Laurel suggest that if it is so difficult to define what ‘interactive’ means, then we are “barking up the wrong tree” as she puts it in attempting to describe what is going on in those terms. I feel much the same way about evaluation. In my opinion, the reason why all existing methods of evaluation are poor at best is that our whole perspective of ‘interacting with a user interface’ (and of assessing the quality of said interaction) is fundamentally the wrong one.

A question which is often raised with respect to evaluation is “other areas of design have no problem with quality testing; why does HCI seem to have such problems?”. It is true that for example car-makers include quality testing processes into the design process with no apparent problem. But if you examine what features car makers are testing, it becomes clear that most of the metrics obtained have only a peripheral connection with usability. It is fairly straightforward to predict a prototype’s fuel economy, or its maximum speed, or its handling characteristics. While all of these features have an effect on the overall “quality” of the car, other features (such as the ‘character’ of a particular model, for instance) are often just as important.

In my Fiat, for example, the (ergonomic) positioning of the door handles almost entirely “interface” it is inevitable that the methods applied will not do very well.

So what model would I recommend to replace those which exist at present? I don’t know — yet. This whole question of how to view the interaction process is one which HCI is still in the early stages of formulating, and there are still many years of work ahead. Some early papers have appeared either addressing the issue of alternate views or else proposing their own: [Kammersgård, 1990] compares and categorizes four different ways of viewing the interaction process, and comes down heavily against what he describes as the “systems perspective” - the conventional view; [Bannon & Bødker, 1991] also highlight the role of perspectives in the tractability of interface problems, and propose an alternative way of viewing the interaction process, based on the Soviet “Human Activity Theory”. This area of study seems at the moment to promise some exciting and useful results in the near future.

10.0 Acknowledgements

I would like to thank my colleagues in COGS for their help and advice in the writing of this paper.Thanks in particular to my supervisors, Steve Easterbrook and Yvonne Rogers, for dedicating many hours listening to me make things up as I went along; to Mike Skaife, for correcting my faulty views on psychology, and to Malcolm McIlhagga who proof-read several drafts, and never stopped badgering me over waffle and bad punctuation. Without his help, this paper would be far less readable. Finally special thanks must go to Judy Delin, who first asked, “OK, can you explain evaluation so that a non-HCI specialist like me can understand it”. Thanks.

Appendix A: The System Usability Scale

For each of the statements below, please circle the box which in your opinion most closely matches your degree of agreement with the statement.

2.	**I found the system unnecessarily complex**	tron Dsa	o w at Dsa r	N t r A r nor Dsa r	o w at A r	tron A r
3.	**I thought the system was easy to use**	tron Dsa	o w at Dsa r	N t r A r nor Dsa r	o w at A r	tron A r
4.	**I think that I would need the support of a technical person to be able to use this system**	tron Dsa	o w at Dsa r	N t r A r nor Dsa r	o w at A r	tron A r
5.	**I found the various functions in this system were well integrated**	tron Dsa	o w at Dsa r	N t r A r nor Dsa r	o w at A r	tron A r
6.	**I thought there was too much inconsistency in this system**	tron Dsa	o w at Dsa r	N t r A r nor Dsa r	o w at A r	tron A r
7.	**I would imagine that most people would learn to use this system**	tron Dsa	o w at Dsa r	N t r A r nor Dsa r	o w at A r	tron A r
8.	**I found the system very cumbersome to use**	tron Dsa	o w at Dsa r	N t r A r nor Dsa r	o w at A r	tron A r
9.	**I felt very confident using the system**	tron Dsa	o w at Dsa r	N t r A r nor Dsa r	o w at A r	tron A r
		tron Dsa	o w at Dsa r	N t r A r nor Dsa r	o w at A r	tron A r

The System Usability Scale is the work of John Brooke of the digital Equipment Corporation, who retains copyright. The SUS may be used freely if its source is acknowledge

11.0 References

Apple, 1987

Apple Computer Inc., *Hu an Int a Gu n App D top Int a* **Addison Wesley, 1987**

Bannon & Bødker, 1991

Liam Bannon & Susanne Bødker, "Beyond the Interface: Encountering Artifacts in Use", in John M. Carrol, ed., *Des n n Intera t on s oo at t e Hu an Co puter Intera e* **pages 227-253. Cambridge University Press, 1991**

ACM SigCHI, 1991

Scott P. Robertson, Gary M. Olson and Judith S. Olson, Editors, *a n rou e no o ro ee n s o AC Con ere e on Hu an Fa tors n Co put n st e s CHI -* **Addison Wasley 1991**

ACM SigCHI, 1992

Penny Bauersfeld, John Bennett, and Gene Lynch, Editors, *tr n a Ba an e ro ee n s o AC Con ere eon Hu an Fa tors n Co put n st e s CHI -* **Addison Wesley, 1992**

Benyon, 1992

David Benyon, "The role of task analysis in systems design", in *Intera t n w t Co puters*, **Vol. 4 No. 1 pages 102-132. Butterworth-Heinemann Ltd., Linacre House, Jordan Hill, Oxford. 1992**

Benyon, 1992, i

[Benyon, 1992], page 115

Brooks, 1972

Frederick P. Brooks, Jr., *e t a an ont Essa s on o t war eEn n eer n* **. Addison Wesley 1975. Reprinted with Corrections January 1982. Especially Chapter 2, pages 13-26.**

Card, Moran & Newell, 1983

Card, S. K., T. P. Moran, and A. Newell *e s oo o Hu an Co puter Intera t on*. **Lawrence Erlbaum, 1983**

Draper, Gray, Kilgour & Oatley, 1991

Draper, S., Gray, Kilgour, Oatley, *Des n o s n nt era t v e v sua nt er a s*, **video 2 (of 5), "Assessing interaction", printed course notes. Department of Psychology, University of Glasgow.**

Diaper, 1989

Diaper, D., Ed. *as Ana s s or Hu an Co puter Intera t on*. **Ellis Horwood, 1989**

Finlay & Harrison, 1990

Janet Finlay & Michael Harrison, “Pattern Recognition and Interaction Models”, in D.Diaper et al., Eds., *Hu an Co put er Int era t on ro ee n s o Int era t* **, pages 149-154. Elsevier Science Publishers (North-Holland), 1990.**

Gray et al., 1992

Wayne Gray, Bonnie E. John, and Michael E. Atwood, “The Precis of Project Ernestine, or, An Overview of a Validation of GOMS”, in [ACM SigCHI, 1992], pages 307-312

Howard & Murray, 1987

Steve Howard and Dianne Murray, “A Taxonomy of Evaluation Techniques for HCI”, in Proceedings INTERACT’87

Jeffries et al., 1991

Robin Jeffries, James R. Miller, Cathleen Wharton and Kathy M. Uyeda, “User Interface Evaluation in the Real World: A Comparison of Four Techniques”, [ACM SigCHI, 1991], pages 119-124

Jeffries & Desurvire, 1992

Robin Jeffries & Heather Desurvire, “Usability Testing vs. Heuristic Evaluation: Was there a contest?”, in *IGCHI Bu e n***, vol. 24, #4, pages 39-42. ACM PRess, 1992**

John & Vera, 1992

Bonnie E. John and Alonso H. Vera, “A GOMS Analysis of a Graphic, Machine-Paced Highly Interactive Task”, in [ACM SigCHI, 1992], pages 251-258

Kammersgård, 1990

John Kammersgård, “Four Different Perspectives on Human-computer Interaction” in Preece, ed., *Hu an Co put er Int era t on s ee te e ra n s*

Kirakowski et al., 1992

J Kirakowski, M Porteous & M Corbett, “How to use the Software Usability Measurement Inventory: The Users’ View of Software Quality”, in *ro ee n s Europ ean Con er n e on o tware ua t* **, Madrid, 1992.**

Macleod,1992

Miles Macleod, “An Introduction to Usability Evaluation”, National Physical Laboratory Report DITC 102/92

Macleod and Rengger, 1993

Miles Macleod and Ralph Rengger, “The development of DRUM: A Software Tool for Video-assisted Usability Evaluation”, unpublished draft, National Physical Laboratory.

Moran, 1981

Moran, T.P., “The Command Language Grammar: a representation for the user interface of interactive computer systems”, in *International Journal of Man-Machine Studies***, 15, 3-50.**

Nielsen, 1992

Jakob Nielsen, “Finding Usability Problems through Heuristic Evaluation”, in [ACM SigCHI, 1992]

O’Donnell et al. 1990

Paddy O’Donnell, Geoff Scobie & Isobel Baxter, “The Use of

Rowley & Rhoads, 1992

David, E. Rowley & David G. Rhoads, “The Cognitive Jogthrough: A fast-paced User Interface Evaluation Procedure”, in [ACM SigCHI, 1992].

Thimbleby, 1990

Harold Thimbleby, *s[illegible] Int[illegible]a [illegible]D[illegible] n*, **Addison Wesley, 1990. Page 130.**

Wharton et.al, 1992

Cathleen Wharton, Janice Bradford, Robin Jeffries and Marita Franzke, “Applying Cognitive Walk-throughs to more Complicated User Interfaces: Experiences, Issues and Recommendations”, in [ACM SigCHI, 1992]

Wright & Monk, 1991

Peter C. Wright & Andrew F. Monk, “A Cost-Effective Evaluation Method for Use by Designers, *Int-J-[illegible] an[illegible] a [illegible] n[illegible] tu [illegible]s*, **35, 891-912**