Empowering Usability Engineering in the Development of Office Products

D. A. TYLDESLEY
User Interface Group, Integrated Office Systems Group, Digital Equipment Company Limited, Imperial Way, Reading, Berks RG2 0SU

Usability engineering is the process whereby the usability of a product is specified quantitatively, and in advance. Then, as the product is built, testing takes place to see whether the planned-for levels of usability have been achieved. This paper describes the way that usability engineering has been implemented by a large engineering group building office software systems. The methods employed are described, and the measures that are taken. By means of two hypothetical product scenarios, it is demonstrated how usability testing is performed and the success that is achieved in feeding back early information to the software engineers. This feedback results in the improvement of the usability of the developing product. The paper ends with some guidelines for successful incorporation of usability engineering into the product development cycle.

Received May 1988

1. INTRODUCTION

Usability engineering is a process whereby the usability of a product is specified quantitatively, and in advance. Then as the product itself, or early 'baselevels' or prototypes of the product are built, it is demonstrated that they do indeed reach the planned levels of usability.

This paper describes briefly the steps traditionally involved in usability engineering in Digital, which are: defining the planned levels through metrics, setting and agreeing these levels, evaluating the product and incorporating user feedback, and designing iteratively until the planned levels are achieved.

These ideas, and Digital's implementation of them are already quite widely published — see for example, Bennett,1 Wixon and Whiteside2 and Brooke.2 Gradually, with increasing experience of employing these techniques, we have found it necessary to adapt them according to the nature of the products that we have to evaluate.

This paper demonstrates how the evaluation techniques vary according to the feedback that was required, and could be used, at different points in the product development cycle. It is important to note from the outset that within a development environment, highly valid measurements of 'usability' are probably less important than the fact that user trials of the product are, indeed, conducted during the development cycle.

2. BACKGROUND

The User Interface Group makes use of usability engineering. The group forms part of the development group that builds and maintains one of Digital's major office products. As such, the User Interface Group is part of Digital Engineering, and is dedicated to one product set. This contrasts with other human factors groups, who may be less directly associated with any one product, but work on a wide variety of products as required. We find this to be an advantage in that it allows us to develop a consistent relationship with the developers, from one product release to the next. We are able to work with them from the early design of the product.

3. HUMAN FACTORS IN PRODUCT DEVELOPMENT

3.1 The engineering process

Product development in Digital follows, more or less rigidly, a series of 'phases'. In the first phase — setting strategy and requirements — the User Interface Group are responsible for producing, and getting agreement on, the User Interface Specification of the group's new products. In the second phase — planning and preliminary design — we produce detailed descriptions of what the screens will look like, how the software will behave for the user and how the keyboard and any pointing device will be mapped to the software.

In the third, implementation, phase, we work with the developers to build the screens, and modify them as necessary to conform to a well-defined User Interface Standard for our set of products. This helps to ensure consistency in the 'look and feel' of the user interface.

It is in the design and implementation phase that evaluation of the user interface usually starts, and proceeds through into the implementation and qualification phases. The latter phase involves internal quality control, and internal and external field testing.

3.2 Usability engineering in the product development cycle

In order to build the user interface with known and planned-for levels of usability, the engineering concepts of measurement, building to the specification and iterative testing are applied. To achieve the necessary iterative design, human factors engineers are involved in three major ways during the design and development process: we help specify the usability goals for the product; we participate in the design of the product; we evaluate the product to ensure that the usability goals are met.

4. SETTING THE USABILITY GOALS

4.1 Cooperation

It is essential that the usability goals be set in cooperation with other groups, such as marketing, product management and the software developers.
Development of the product may have a variety of constraints on it that are not directly concerned with the usability of the product. For example, it must be designed so that users of another product will find it easy to use, or there may be a large installed base of existing users, for whom the new version must not prove disruptive, or the product must ship within 2–3 months of another product, or the product must be easy to translate, and so on.

Any usability goal set by the human factors engineer must take these other requirements into account. Failure to do so will result in unrealistic requirements. Participation with the development team in setting the usability goals means that the usability goals are less likely to get ‘traded-off’ against other product requirements in the competition for ‘scarce’ engineering resources.

4.2 An operational measurement-based definition

An operational usability definition establishes a common meaning for usability throughout the development team. Product requirements in the past used to state vaguely ‘the product will be easy to use’, but as Brooke6 observes ‘as a requirement for the project manager to meet, it is virtually useless’. If usability is stated in clear, measurable terms, and the project team have participated in setting those criteria, then it is likely that a good attempt will be made to achieve the goals.

An operational definition of usability focuses the design team’s efforts in to a limited, rather than an open-ended definition. As has been wryly observed, ‘part of the problem is the continuing battle between development managers and special interest groups in the development team, e.g. performance, software architecture, quality, documentation, human factors’.

Imagine a group of acoustic engineers instructed to make ‘a quiet terminal’, compared to a second group told to design ‘a terminal that produces no more than 65 decibels, measured in an anechoic chamber’. It’s the same principle in usability engineering. The operational definition should be written, open to public inspection and modification. It should be communicable, and have the same meaning to different people on the project and to the users of the product. Metrics should be specified that can be used to judge ease of use and target levels for that metric put into the product requirements.

The company templates for gathering product requirements always contain a usability section, and we are frequently asked to submit this section for new products. Increasingly, management and software engineers also contribute to the usability sections. This may be as a result of the increasing emphasis on the user interface that accompanies the design of products with high-resolution, direct-manipulation user interfaces.

So, what metrics can we use?

4.3 Performance measures

First, we need to take into account objective or performance measures of usability – can people achieve useful work in a reasonable time using the system? For example, Brooke6 describes the Work-Speed Metric, which has been used for development of a number of office products. I will not dwell on it here – suffice it to say that it gives a reasonable measure of work rate on a specified ‘benchmark’ (laboratory) task. It permits comparisons to be made across products for the same benchmark task and it is computed using the time spent on the task, the percentage of the task completed and an arbitrary constant based on the fastest possible task solution for a practised expert user.

This may not be the optimal measure. It obviously doesn’t take all the possible aspects of efficiency of use into account, such as frequency of errors and confusions. But decisions are often made on far less valid grounds. And, in the development situation, in the absence of any better metric, this will be acceptable.

Whiteside et al.5 give the following checklist of criteria for usability. Such a list is still not exhaustive. As they stress, ‘Use these tables as a starting point. Remove items inappropriate to your situation, and add new ones. Make them your own by incorporating items relevant to your work context and the user data you can collect.’

4.3.1. Possible measurement criteria

(1) Time to complete task
(2) Percentage of task completed
(3) Percentage of task completed per unit time (speed metric)
(4) Ratio of successes to failures
(5) Time spent in errors
(6) Percentage number of errors
(7) Percentage or number of competitors that do this better than current product
(8) Number of commands used
(9) Frequency of help or documentation use
(10) Time spent using help or documentation
(11) Percentage of favourable/unfavourable user comments
(12) Number of repetitions of failed commands
(13) Number of runs of successes and of failures
(14) Number of times the interface misleads the user
(15) Number of good and bad features recalled by users
(16) Number of available commands not invoked
(17) Number of regressive behaviours
(18) Number of users preferring your system
(19) Number of times users need to work around a problem
(20) Number of times the user is disrupted from a work task
(21) Number of times the user loses control of the system
(22) Number of times the user expresses frustration or satisfaction

(Extracted from Whiteside.5)

4.4 Attitude measures

Secondly, we need to consider subjective, attitudinal measures of usability – did the user find the product pleasing, straightforward and consistent to use? This is a much more subtle concept than simple performance measures. Users may like a product even through they are not able to achieve much with it, or they may dislike it heartily, even though they can work fast and efficiently with it.

We have used with success a 10-item rating scale containing items such as ‘I think I would like to use this system frequently’ and ‘I think I would need the support of a technical person to use this system’. Users indicate
the extent to which they agree or disagree with each item, on a 5-point scale. The list consists of 5 negative and 5 positive points. The overall usability rating is computed as the sum of the item ratings, with the ratings for the negative items inverted. The score is then normalised so that it lies on a scale of 0–100. Hence a rating of 50 is a fairly neutral rating of the usability of the system; above 50 indicates a positive attitude.

This System Usability (SU) rating scale is perhaps our most successful and widely applied usability metric. A body of knowledge relating to perceived usability on a wide range of products has now been established. In addition, a similar System Consistency (SC) scale has been developed to examine the user-perceived consistency of two products that are supposed to have a ‘consistent user interface’. This is built up of questions such as ‘Knowing one system would help me learn the other more quickly’ – an indicator of ‘performance’ consistency – and ‘These systems are similar in appearance’ – an indicator of ‘aesthetic’ consistency between two products.

4.5 The usability specification

Usability components can best be represented in a usability specification. This format gives an overview of all the operationally-defined criteria for desired system attributes. Seven usability items are defined for each system attribute: measuring concept, measuring method, worst case, planned level, best case, and the current level. Such a specification can constitute the publicly agreed usability goals that are used in the product development cycle. While constructing them, we must ask ourselves the questions:
- are the attributes measurable?
- are the users clearly enough specified?
- are there resources to measure all the attributes?
- do all project members agree on each attribute?
- how well do the attributes capture the usability of the system?

In practice, we have found that a set of general statements of usability is also important to set the scene for the specific requirements, for example: ‘It must be possible for an experienced end-user of System A to do System Administration on System B’. Sometimes it is not possible to measure all of the attributes. None the less, the very process of establishing the specification gets people thinking and talking about the usability of the product early on in the development cycle. In this alone, the specification has served one of its purposes.

5 PARTICIPATE IN THE DESIGN AND BUILD OF THE PRODUCT

From what I have said above, it should be clear that the structure of our group allows us to work very closely with the developers. This is important, because no product is ever predictable, and there are always issues which impact the usability, that arise only during implementation.

We find we have to continuously modify the specification of the user interface. We must also be prepared to do some code-level work ourselves; for example, we modify the screens and some parts of the software, such as high-level scripts. In addition, it is necessary to attend design team meetings to be a voice for the user when other constraints are being voiced. In these meetings we depend heavily on information gathered from user evaluations of prototypes and earlier versions of the product. This brings me to the final section – evaluating the user interface.

6. EVALUATE THE USER INTERFACE

6.1 Importance

The user interface of the product will need to be debugged, like any other part of the software. Also, our participation in the design process will not guarantee that the best design option will be chosen. It is possible, often probable, that things will happen in the development of the product that will cause other design assumptions to become invalid. So the product will need to be evaluated with representative users, during the development process. These user evaluations are intended to establish that the usability goals have been achieved and identify any changes that are required to improve the overall usability of the product. One method of identifying changes that are necessary, and also ranking them in order of importance is called Impact Analysis. This method, based on the work of Gilb (1977) identifies user problems and shows the impact of each problem on the total user problem time. Development effort can then be accordingly concentrated on the main areas of user difficulty. Further details on this method can be obtained from Good et al.4

If the usability goals are not achieved, re-design of the user interface may be necessary. The decision to do this is easier because the goals have been established in advance and the human factors engineer has participated throughout the design cycle. We are no longer in the situation where the human factors engineer is called in after the product has been built, to ‘rubber stamp’ the product for usability.

6.2 Methods

The methods adopted for evaluating the product will vary widely with the product in question and the metrics established for usability. Whiteside et al. suggest that the following checklist of measurement operations should be considered. Again, this is not an exhaustive list, and the appropriate methods will need to be selected on the basis of factors such as the time available, the type of users required and the nature of the product.

(i) Ask users to perform a specific task.
(ii) Monitor users during free use (logging and/or observing).
(iii) Give users a questionnaire.
(iv) Interview users.
(v) Survey users.
(vi) Ask users for critical incidents revealing successes or failures.

6.3 The evaluation plan

With these objectives in mind, an Evaluation Plan is drawn up, which can be summarized by the following checklist of decisions: Scope; Intended Users; Metrics; Research or Development?; Methods.
6.3.1 Scope
Here we must consider whether we are testing the whole system, a component part or a prototype. Testing a whole system may mean that only certain parts of the system can be exercised in the time available. In such circumstances, we may decide to guide users to particular aspects of the system, log their usage, and look for critical incidents and comments on the user interface. If a component part of the system is being tested, for example the Calendar or the Index, then more formal ‘benchmark’ methods can be used. Testing prototypes permits us to construct formal experimental designs and increase the rigour of our testing. Working prototypes, on the other hand, usually need more preparatory work, to ensure that they work in a realistic fashion.

6.3.2 Intended users
Here we consider the type of users for whom the product is designed; are they initial level, casual users, experienced users or existing users of another system; are they customers or internal employees? Representative users of the product are best found on customer sites, though quick evaluations may mean that we are forced to employ internal users. External users may need to sign ‘non-disclosure’ forms regarding prototypes and products that they test. Experienced users of office systems are becoming more readily available.

6.3.3 Metrics
Here the decision has to be made whether quantitative or qualitative results are required. For example, a new indexing mechanism to the user’s file cabinet may be under examination. Simple timings on a benchmark task, the work-speed metric and system usability ratings could all be achieved quite quickly; the whole investigation, including set-up of the benchmark task, and analysis and report of findings can usually be achieved in 2/3 weeks. When testing whole systems, under constraints of limited time, observation of users, interview and analysis of comments and opinions are usually sufficient. When time is not a constraint, for example after product release, the more time-consuming impact analysis is usually conducted. This generally means that required changes to the user interface that are identified by impact analysis cannot be implemented until the next release of the product.

6.3.4 Research or development?
Whether the evaluations are part of long-term development of future products or development of products for customer release in the near future will affect the time available to run the trials, the use that is made of the feedback, and whether the trials have to be conducted under conditions of non-disclosure.

6.3.5 Methods
These are usually determined by the factors mentioned above – for example, the chosen metrics and whether the evaluation is related to development of a specific product or is research for a future product. The methods employed include ‘benchmark’ testing, user interviews, ‘walkthroughs’ of the software combined with interviews, observation and trial of prototypes. The choice of laboratory trial or customer site visit will also affect the technique chosen. The decision whether to videotape the user sessions or not must be made. No firm guideline can be given to decide on the methods employed – each evaluation tends to be judged as a separate case.

6.4 Two sample scenarios
The variability in methods can be illustrated by reference to two hypothetical evaluations. These are (1) a formal evaluation with benchmark task and an impact analysis, (2) a less formal evaluation or ‘walkthrough’ of a new product user interface.

6.5 Formal method

6.5.1 Objectives
The objectives of this evaluation are to establish usability measures for ‘System B’ for comparison with other office systems, and to see how consistent this system is with an existing office system, ‘System A’. An evaluation plan is drawn up which decides on the goals shown in Table 1.

6.5.2 Analysing the data
Four types of data are extracted from this evaluation: problem codes; the Work-Speed Metric; the System Usability Rating; and the System Consistency Rating.

Problem codes. Careful analysis of the videotaped sessions is carried out to identify areas where the user interface confuses or misleads the users. These problems are described and given a code (see Table 2).

<table>
<thead>
<tr>
<th>Table 1. The Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Factor</strong></td>
</tr>
<tr>
<td>Scope</td>
</tr>
<tr>
<td>Intended users</td>
</tr>
<tr>
<td>Metrics</td>
</tr>
<tr>
<td>Res/Devt?</td>
</tr>
<tr>
<td>Methods</td>
</tr>
</tbody>
</table>
Table 2. Coded Problem

<table>
<thead>
<tr>
<th>Problem title</th>
<th>Problem description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETURN at Index</td>
<td>Users attempt to get a listing of items by pressing RETURN at the appropriate field on an index form. Instead this simply moves the cursor to the next field, and lists after the cursor leaves the last field</td>
</tr>
</tbody>
</table>

Typically in an evaluation, 20/30 user problems are identified in this way, which contribute to a substantial delay in the user’s work. These are ranked, based on their individual contribution to the total problem time. This ranking (impact analysis) is used to decide which problems should be corrected first. The listing of problems in this brief fashion is usually the most effective output from the evaluation.

*Work/Speed Metric*. This is computed to give an indication of the user productivity on System B.

*System Usability Score*. This is useful to give an indication of the users’ general attitude to the ease of use of the system. For example, we might note a negative transfer of users from System A to System B. (This may be a temporary effect of the switch from one system to the other, but even so it is important, because these may be the very users for whom the product is intended.)

*System Consistency Score*. The results of this questionnaire indicate whether the users rated the two systems (‘A’ and ‘B’) as reasonably consistent, for example a score of 61 shows the two systems to be reasonably consistent; a score of 50 represents a neutral attitude.

### 6.6 Less formal method

#### 6.6.1 Objectives

In this second (hypothetical) evaluation. ‘System C’ is an office system with a large installed base of users. Changes to the user interface must lead to minimal disruption of their normal working patterns when a new version is installed. At the same time, users must perceive the new version to be more usable than previous versions.

Evaluations are performed to examine some major changes that had been made to the user interface of a new version. In contrast to the first example evaluation, this must be done quickly, between the time that the code first becomes available, and while there is still time to modify the code. Again, an Evaluation Plan is drawn up, as shown in Table 3:

#### 6.6.2 Analysis of Data

In contrast to the first example, only coded problems, customer comments and attitude ratings can be obtained. The list of coded problems is generated and worked upon in the same way as the formal evaluation, but the problems are no longer ranked for importance. The presence of a researcher makes measurement of problem times impossible, as he or she will answer user queries and stop them before they spend too long doing something that is incorrect.

This is a useful method when time is limited. Users clearly prefer the presence of a researcher in the room with them when completing the benchmark task. The presence of the researcher to give hints and tips, and to steer the user to areas of the user interface that were new, obviously means that impact analysis is no longer valid. The problem areas can still be identified – but we do not know how long the user might have become stuck in a particular problem.

System usability and system consistency questionnaires can still be filled in after the walkthrough of the benchmark Task. These scores tend to be higher than normal on a new version. This is probably due to the hints and tips given, which help the user to avoid the pitfalls, and thereby create a better impression of the product. An advantage of this method is the richness of the discussion recorded. During informal trials, we depart from the normal method of reviewing the videotapes after the trials – this loses a lot of information and takes immense amounts of time. Instead, we log the discussions directly into electronic files. These electronic files are then edited and ‘tags’ inserted at relevant comments. These comments usually fall into one of 30 or so different categories. By making use of list processing, a part of the office system itself, we can separate out comments on specific topics in a very short space of time.

A further aid to provide feedback from the evaluation quickly and efficiently is a videotape editing suite, which allows us to put together a set of video clips of the problem areas in very short time. This is very effective in

### Table 3. The Plan

<table>
<thead>
<tr>
<th>Factor</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope</td>
<td>The new ‘baselevel’ of System C is to be installed and tested. System C is still being built</td>
</tr>
<tr>
<td>Intended users</td>
<td>These are experienced users of System C, but an older version. They are customers. As large a sample size as possible is required</td>
</tr>
<tr>
<td>Metrics</td>
<td>SU, SC, customer comments</td>
</tr>
<tr>
<td>Res/Dev?</td>
<td>Direct input to development team. Time is crucial – 4 weeks only before documentation is ‘frozen’. The product is subject to non-disclosure to customers</td>
</tr>
<tr>
<td>Methods</td>
<td>Benchmark task modified by a researcher ‘sitting-in’ to answer questions and to give hints; videotaping, laboratory-based trials</td>
</tr>
</tbody>
</table>
convincing other project members of the need to make some user interface changes.

It is small advances such as this, and the speed with which feedback could be given to the development team, that make this informal ‘walkthrough’ technique a particular success.

7. SUMMARY
We are gradually improving our techniques of evaluation and feedback of results, to suit the software development environment. The following basic principles have been demonstrated in this paper, and guide our work.

- Get everyone to agree what they really mean by ‘easy to use’ and write it down (the usability goals).
- Work with developers to ensure that you understand their decisions and that they understand the needs of users.
- Use a range of measures, subjective and objective, to test if the usability goals are met.
- Be prepared to modify your evaluation techniques to suit the product, the time available, and the users.

- Use any tools possible to extract the feedback quickly and provide it to the developers.
- Do not underestimate the richness of the information that can be derived from just talking to users.

The methods outlined in this paper are under continuous review and development. One particular area currently being focused upon is that of increasing the ecological validity of the methods used. All the examples given here rely upon inviting users to work on the developing products in a ‘laboratory’ environment. More effort will be made in the future to transfer these methods and prototype software to customer sites, where users performing ‘normal’ tasks of their choosing can be used to evaluate the products. To do this, the methods known as ‘contextualised interviews’ are likely to come to the fore.5

Acknowledgements
The author would like to thank Bob Griffin for his comments on an earlier draft of this paper, and John Brooke and John Whiteside who provided the original stimulus for the methods described in this paper.

REFERENCES