Designing and testing the usability of interactive software with formative and summative evaluations

Matthias Rautenberg

Work and Organizational Psychology Unit, Swiss Federal Institute of Technology (ETH)
Nelkenstr. 11, CH-8092 Zürich, Switzerland, email: rautenberg@rzvax.ethz.ch

1. A CLASSIFICATION OF EVALUATION METHODS

One of the main problems of standards (ISO, DIN, etc.) in the context of software ergonomics (HCD) is, that they can not be measured in a quantitative way (see [15]). The scope of this position paper encluses the product "interactive standard software." Today there are four different views on human computer interaction to measure interactive qualities (see [1];593, [3], [15]).

1. The interactive oriented view: usability can be measured by examining how the user interacts with the product (-> formative and summative evaluations [11]); this view is the most common one; all kinds of usability testing with "real" users and "real" computer systems are subsumed here [8] [11].

2. The user-oriented view: usability can be measured in terms of the mental effort and attitude of the user (-> questionnaires and interviews);

3. The product-oriented view: usability can be measured in terms of the ergonomic attributes of the product (-> quantitative measures); all heuristic evaluations carried out by ergonomic experts investigating a concrete software product fall in this category.

4. The formal view: usability can be formalized and simulated in terms of mental models (-> formal concepts); Karal [7] describes the formal methods in the context of "theory-based" evaluation; the limitations of this view are discussed by Ziegler ([12];231ff).

These four different views can be classified by the two dimensions "user" and "computer", which are really or virtually involved in the evaluation (see Table 1). Till today nobody knows, how these four different views can be successfully combined (see the results of Jeffries et al. [6] and the discussion by Jeffries and Desurvire [5]). Nevertheless, there are invariable errors or misconceptions which will be discovered in any process of having users test a system that even the most insightful of designers would miss. It is a necessary experience to become aware of the fact that none of us knows everything about how the human mind functions" (7); 895.

Table 1

A classification scheme for the four different views on human computer interaction to measure interactive qualities.

<table>
<thead>
<tr>
<th>User</th>
<th>virtual</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>form</td>
<td>virtual</td>
<td>user</td>
</tr>
<tr>
<td>oriented</td>
<td>real</td>
<td>interactive</td>
</tr>
</tbody>
</table>

2. USABILITY TESTING OF INTERACTIVE SOFTWARE

Simple and fast techniques for involving users include discussion groups with various communication aids (metaphor, layout sketches, "screen-dumps", scenarios, etc.), questionnaires for determining the attitudes, opinions and requirements of the users, the "walk-through" technique for systematically clarifying all possible work steps, as well as targeted interviews aimed at a concrete analysis of the work environment [4] [10] [20] (see also Table 2).

Very sound simulation methods (e.g., scenarios, "wizard of oz" studies) are available for developing completely new systems without requiring any special hardware or software. Spencer [19] presents a summary of techniques for the analysis and evaluation of interactive computer systems. Comparative studies, e.g., summative evaluations and interactive benchmark tests [14], can be undertaken after the second time through, when working with a version concept, for then there are at least two versions available.

In an interdisciplinary research project we developed and tested different methods to involve users in the design of interactive software. Our involvement in the development of a
relational database system gave us the possibility to develop methods for user participation in the case of standard software development, too. Figure 1 shows an overview over all interactive usability tests during three development cycles. The results of the most important effects are published elsewhere (see [12] [14] [17]).

**VERSION** | **YEAR** | **ACTIVITY** | **RESULT**
--- | --- | --- | ---
[1.0] | 1. year | desktop interface | desktop interface
[1.0] | 2. year | test results on development as a test | test results on development as a test
[1.0] | 3. year | test results on development as a test | test results on development as a test
[1.0] | 4. year | test results on development as a test | test results on development as a test

Figure 1. An overview over all interactive benchmark tests during three development cycles.

3. AN ITERATIVE CYCLIC PROCESS MODEL

The literature contains a wide series of suggestions for incorporating cycles into software development procedures [3] [4] [10] [14]. The type of software to be developed has proved to be one of the essential factors governing software development. The following four types can be distinguished [17]:

- **Type A**: Specific application for an internal division; both the division placing the order and the one developing the software belong to the same company.
- **Type B**: Specific application for external users; the division placing the order and the one developing the software belongs to different companies.
- **Type C**: Standard solutions for external users: this often arises from projects of Type A or Type B, when individual software solutions (Type A, Type B) are specially adapted for further users.
- **Type D**: Standard software for an anonymous user group.

The global optimization cycle begins at Quadrant-I (Figure 2) when developing completely new software and at Quadrant-IV in the case of further development and refinement of existing technology. The global optimization cycle together with its incorporated local cycles, can be subdivided into four regions (Quadrants I – IV of Figure 2).

Quadrant-I includes the analysis and approximate specification. Communicative, informal methods are mainly applied here. Detailed specifications are optimized in Quadrant-II using prototypes. The specified hardware and software are implemented in Quadrant-III and the test data assessed. Quadrant-IV comprises assessment, maintenance and optimization of the system in the real-life operating environment. The effort spent on optimization in each quadrant varies according to the type of the projects and of the type. However all software development project analyses completed to date indicate that increasing the effort expended on optimization in Quadrant-I reduces maintenance in Quadrant-IV and saves costs [2] [10] [17].

We present here an iterative-cyclic software development concept that integrates solution proposals developed to date for overcoming the specification, communication and optimization barriers on the basis of the notion of an optimization cycle [15]. The global optimization cycle can be subdivided into four regions: the region where requirements are determined (Quadrant-I), the region of specification (Quadrant-II), the region of implementation (Quadrant-III) and the region of application and maintenance (Quadrant-IV).

Different aspects of the work system to be designed can be progressively optimized as one moves from quadrant to quadrant. The various perspectives of the ideal sought take on progressively more concrete form. An appropriate investment in optimization in Quadrants-I and -II not only helps to reduce the total cost (development costs and application costs), but also lead to optimally adapted hardware and software solutions. This is because all subsequent users are involved at least through representatives, and can therefore inject their relevant knowledge into the design of the work system. User involvement can be done by different methods and techniques (see Table 2),

---

In: R. Taylor & J. Coutaz (Eds.), Preprints of Proceedings of 'Workshop on Software Engineering and Human-Computer Interaction: Joint Research Issues' (pp. 183-186). Irvine (USA): University of California
Figure 2. An iterative cyclic process model with user participation [13].

<table>
<thead>
<tr>
<th>Method</th>
<th>Action</th>
<th>Test</th>
<th>Outcome</th>
<th>Cycle-Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discussion-I</td>
<td>Verbal communication</td>
<td>Verbal interpretation</td>
<td>Global design decisions</td>
<td>Seconds - minutes</td>
</tr>
<tr>
<td>Discussion-II</td>
<td>Mota-plan, Flip-charts, etc.</td>
<td>Visual &amp; verbal interpretation</td>
<td>Specific design-decisions</td>
<td>Minutes - hours</td>
</tr>
<tr>
<td>Simulation-I</td>
<td>Sketches, scenarios,</td>
<td>Visual &amp; verbal interpretation</td>
<td>Specification of the interface</td>
<td>Minutes - days</td>
</tr>
<tr>
<td></td>
<td>&quot;Wizard of Oz&quot;, etc.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulation-II</td>
<td>Drafting of structural</td>
<td>Visual &amp; verbal interpretation</td>
<td>Specification of the informal descriptive</td>
<td>Hours - weeks</td>
</tr>
<tr>
<td></td>
<td>blueprints, etc. with</td>
<td></td>
<td>documentation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(semi-)formal methods</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototyping-I</td>
<td>Horizontal prototyping</td>
<td>&quot;Thinking aloud&quot;,</td>
<td>Specification of dialog component</td>
<td>Days - weeks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Walk-through&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototyping-II</td>
<td>Partial vertical prototyping</td>
<td>Heuristic evaluation</td>
<td>Partial specification of application component</td>
<td>Days - weeks</td>
</tr>
<tr>
<td>Prototyping-III</td>
<td>Complete vertical prototyping</td>
<td>Task-oriented benchmark</td>
<td>Specification of application component</td>
<td>Weeks - months</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mark tests</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Versions-I</td>
<td>Run through entire</td>
<td>Summative evaluations</td>
<td>First fully complete version</td>
<td>Months- years</td>
</tr>
<tr>
<td></td>
<td>development cycle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Versions-II</td>
<td>Run through entire</td>
<td>Summative evaluations</td>
<td>Several fully complete versions</td>
<td>Months- years</td>
</tr>
<tr>
<td></td>
<td>development cycle</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Acknowledgments: The preparation of this paper was supported by the German Minister of Research and Technology (BMFT AuT program) grant number 01 HK 065-3. As part of the BOSS "User oriented Software Development and Interface Design" research project.

I have to thank following persons: Dr. P. Spinas, Dr. K. Schlaghauer, O. Strohm, R. Mollenhauer.

185

In: R. Taylor & J. Coutaz (Eds.), Preprints of Proceedings of 'Workshop on Software Engineering and Human-Computer Interaction: Joint Research Issues' (pp. 183-186). Irvine (USA): University of California
We used systems theoretical considerations to explain how each optimisation cycle consists of a test and an action component that are suitably coupled. Each component can be of a widely varying nature. Table 2 provides a survey of the main focus of effort, of the nature of activity, and of the tests, the outcome and the expected range for the length of the cycle. The shorter an optimisation cycle is, the more rapidly — and therefore the more often — it can be used to reach a truly optimal result.

4. CONCLUSIONS

The dominance of technical aspects in conventional interface design led to insufficient usability of interactive software by end-users. Taking the user’s task oriented view seriously leads to new, useful interface concepts (see 14). As more the effort expended on optimization in the first quadrants, so less is needed in Quadrant IV. The amount of effort required for optimization in the second and third quadrant depends on the complexity of the system to be designed. The investment in Quadrant-II can be minimized for example with the help of modern prototyping tools and specification methods that the user can easily understand. Employing powerful development environments and suitably qualified software developers will improve the investment in Quadrant III. But first, we must start learning to plan jointly technology, organization and the application of human qualification.

5. REFERENCES

Workshop on Software Engineering and Human-Computer Interaction: Joint Research Issues

Preprints

UNIVERSITY OF CALIFORNIA, IRVINE

May 16 — 17, 1994

For more information please contact either of the workshop co-chairs:

Richard N. Taylor, University of California Irvine
Email: taylor@ics.uci.edu, Fax: +1-714-856-4056

Joëlle Coutaz, IMAG Grenoble
Email: joelle@imag.fr