ENCyclopedia OF MICROCOMPUTERS

EXECUTIVE EDITORS
Allen Kent  James G. Williams
UNIVERSITY OF PITTSBURGH
PITTSBURGH, PENNSYLVANIA

ADMINISTRATIVE EDITORS
Carolyn M. Hall  Rosalind Kent
PITTSBURGH, PENNSYLVANIA

VOLUME 19
Truth Maintenance Systems
to Visual Display Quality

MARCEL DEKKER, INC.  NEW YORK • BASEL • HONG KONG
CONTENTS OF VOLUME 19

Contributors to Volume 19

TRUTH MAINTENANCE SYSTEMS
Neli P. Zlatareva

THE TSQL2 BASELINE CLOCK
Curtis Dyreson and Richard T. Snodgrass

AN UNCERTAINTY PRINCIPLE FOR INFORMATION SEEKING:
A QUALITATIVE APPROACH
Carol C. Kuhlthau

USABILITY ENGINEERING
Keith A. Butler

THE USE OF AN ANIMATED TUTOR IN TEACHING CHINESE
Hua-Yuan Li Mowry

USE OF AI-BASED APPROACHES FOR DETERMINING CNC
MACHINING PARAMETERS IN MANUFACTURING
Kyung Sam Park and Soung Hee Kim

Diane Nahl

USER INTERFACE EVALUATION
Harold Reiterer

USER INTERFACE SOFTWARE TOOLS
Brad A. Myers

USER MODELING AND GRAPHICAL USER INTERFACE
DESIGN FOR COMPLEX SYSTEMS
Gorilla A. Sandström

USER MODELS IN ADAPTIVE SYSTEMS
Sandra Corberry and Jennifer Chu-Carroll

v
1
49
69
83
113
129
143
201
223
259
275
USER INTERFACE EVALUATION

INCREASING NEED FOR USER INTERFACE EVALUATION METHODS

User interfaces are becoming critical to the success of software products. However, designing and evaluating user interfaces is difficult and time-consuming. There are many reasons why a focus on the user interface is important and why user interface design and evaluation are inherently difficult tasks (1). The importance of a focus on human–computer interaction has been recognized by industry, academia, and government. The U.S. Committee to Assess the Scope and Direction of Computer Science and Technology of the National Research Council in their report “Computing the Future” lists user interfaces as one of the six “core sub fields” of computer science, and notes that it is “very important” or “central” to a number of important application areas such as global change research, computational biology, commercial computing, and electronic libraries (2). The reasons for the increasing importance of user interfaces are as follows:

• User’s demands on software have changed; they expect to be able to sit down and use software without spending their time reading manuals.
• Empirical results show that attention to usability dramatically decreases costs and increases productivity.
• Poor user interfaces have contributed to disasters, including loss of life.

The benefits of good user interfaces are clear, but there is surprisingly little attention paid to why user interfaces are difficult to design and evaluate. When user interfaces are developed, technical questions are often overemphasized, and organizational and social consequences may be neglected. This often results in hard-to-use and not very transparent user interfaces. The consequences for the employees are frustration and anxiety as well as stress. The consequences for the company are lost flexibility of the organization, absenteeism costs, turnover costs, and poorly qualified employees. Against the background of these consequences, the user interface characteristic called either “user-friendliness,” “ease of use,” “usability,” or “ergonomic design” has become very important. The International Organisation for Standardisation (ISO) defines good ergonomic design of visual display terminal (VDT) work: “...to ensure that VDT users can operate display screen equipment safely, efficiently, effectively and comfortably. In practice, this can only be achieved by careful design of the VDTs themselves, the workplaces and working environments in which they are used and the way the VDT work is designed, organized and managed (3).” This definition is not restricted to user interface design; it includes the design of appropriate application functionality, organizational design, and job and task design. Therefore, user interface designers are faced with different—often conflicting—requirements and need answers to the question, “How can usability requirements be considered and evaluated during application development?” To
consider usability requirements, the designers need design criteria and design rules, appropriate design methods, and design tools. To evaluate usability during and after the development process, they need appropriate evaluation methods that give them feedback on the ergonomic quality of their work. In these terms, usability is an integral part of software quality in general.

Another reason for the increasing need for evaluation methods is the new European Economic Area. To establish common working conditions for VDT users, the European Community (EC) published a "directive concerning the minimum safety and health requirements for VDT workers (4)." The national governments of the EC members have to transform this directive into national law. In this process, the European standardization activities of the CEN (Comité Européen de Normalisation) and the international standardization activities of the ISO concerning ergonomic requirements for VDTs will have great influence. Especially the ISO standard 9241 "Ergonomic requirements for office work with VDTs" plays an important role. In the future, this standard will often be an integral part of a software requirement specification. Therefore, software developers will have to take these requirements and principles into consideration. This also includes conformance testing their products against this standard. On the other hand, software buyers also need evaluation methods to test conformance with the standards. Both groups, developers and buyers, need practical software evaluation methods.

CONTEXT OF USE

The evaluation of usability of the user interface has to be placed in the context of use of an overall system consisting of the user, his or her job and tasks, the hardware and software, and the organizational, technical, and physical environment. ISO 9241 Part II gives the following definition of usability: "Usability is measured by the extent to which the intended goals of use of the overall system are achieved (effectiveness); the resources that have to be expended to achieve the intended goals (efficiency); and the extent to which the user finds the overall system acceptable (satisfaction) (5)."

Although usability is a property of the overall system, the focus of attention is usually on a specific element within the overall system. Within the scope of user interface evaluation, this element is the software product. It is possible to address the usability of the user interface, but only when the particular context of use has been identified.

Figure 1 shows all the elements of the context of use and the relationships between them. The investigation of the elements of the context of use is done by considering the following characteristics (5):

- **The user**: User types (e.g., user populations) based on aspects about users skills and knowledge (e.g., software experience, hardware experience, task experience, organizational experience, education, training), personal attributes (e.g., age, gender, physical capabilities, disabilities), cognitive attributes (e.g., intellectual abilities, attitude, motivation)
- **The software**: Descriptions of the functionality's and main application areas of the software, available instructional items (e.g., handbooks)
rules, and it gives the new DT understanding of process, ergonomic role, elements of production and evaluation.

The job and tasks: Details about the job of the user as a whole, and the tasks for which the software will be used as an aid (e.g., task goal, task name, task frequency, task breakdown, task duration, task flexibility, task output, task dependencies)

Organizational environment: Aspects of the structure of the organization (e.g., hours of work, group working, job function, work practices, management structure, communication structure, interruptions), the attitudes and culture (e.g., policy on use of computer, organizational aims, industrial relations), and the job design (e.g., job flexibility, performance monitoring, performance feedback, pacing, autonomy, discretion)

Technical environment: Hardware and basic software (e.g., operating system) which is necessary to use the software, reference material

Physical environment: Workplace conditions (e.g., humidity, temperature, noise), design of the workplace (e.g., space and furniture, user posture, location), workplace safety (e.g., health hazards, protective clothing, and equipment).

The relationship "accomplish tasks" describes how a user can carry out the tasks. The relationship "interaction" describes how difficult or easy it is for the user to interact with the software. The relationship "functionality" describes how well the software supports the tasks and allows the user to reach the task goals. An evaluation of the user interface has to take all these elements and their relationships into consideration.

SURVEY OF EXISTING EVALUATION METHODS

Today, many evaluation methods are available, but no one is sufficient by itself. Each method has its advantages and disadvantages, as the following classification shows.
Subjective Evaluation Methods

Subjective evaluation methods are directly based on the user's judgment. The user is the source of the evaluation, possibly even its initiator. The user of a system is asked questions about certain system properties. The answers are based on the user's accumulated experience. A distinction must be made between oral and written questions. In addition, the "thinking aloud" method should be mentioned here. In this method, the users perform a task while giving verbal expression to their thoughts, problems, opinions, and so forth, all of which provides the evaluator with indicators for interpreting the test. This approach may seem artificial to the user, and one alternative is the "constructive interaction" method, in which two users work together on a task and "tell" each other what they are feeling, doing or intending to do, and so on. This generates data in a more "natural" manner. Subjective evaluation methods tend to yield subjective ("soft") data (e.g., whether the system is comfortable, pleasant to use, manageable, comprehensible, etc.) rather than objective ("hard") data (e.g., whether a system performs a task quickly and error-free). The advantages of subjective methods are those of low cost, ease of implementation, an ability to pinpoint unstructured problems, and so forth. The drawbacks are a tendency to produce exaggerations, the difficulty in avoiding leading questions, a plethora of data, which makes evaluation a costly matter, and the low regard in which such methods are held by those questioned. Examples of subjective evaluation methods based on written questions and answers that can be practically applied are the "Questionnaire for User Interaction Satisfaction (QUIS 5.0)" (6) or the "Evaluation Checklist" (7). A new and very interesting subjective evaluation approach is the "Software Usability Measurement Inventory" (SUMI) of the MUSIC Project (8).

Objective Evaluation Methods

Within objective evaluation procedures, we find a large number of approaches, ranging from paraexperimential studies (e.g., "Wizard of Oz"), through the evaluation of system properties based on checklists, on up to classical experiments. The advantage of objective evaluation methods is that they are not based on subjective judgments by users or evaluators (no "soft" data). The disadvantage of objective evaluation methods is their limited scope of observation (only "hard" data). For example, in logfiles of user behavior with input media, recorded by the computer itself, observation may be concealed but is necessarily confined to the user's handling of the system. Any other actions or interactions (e.g., signs and gestures, exclamations, use of manuals, communication with others, etc.) are not recorded. A good example of an objective evaluation method is the "Performance Measurement Method for Usability Evaluation" of the MUSIC Project (8).

Expert-Based Evaluation Methods

These methods lie at some intermediate stage between subjective evaluation methods and objective ones. In these methods, a system is examined by an expert. Unlike the user in the question-and-answer sessions discussed earlier, the expert's approach is based less on a task to be performed by the tested system than on questions asked by software ergonomics. These methods are subjective because the expert examines
and answers questions asked by software ergonomics according to his personal assessment. They are objective because the examination criteria of software ergonomics are operationalized and precisely formulated to an extent enabling the evaluator to answer questions based on clear test rules and traceable conditions. The advantages are that the expert judgment is relatively fast, uses few resources, provides an integrated view, and can address a wide range of behavior. On the other hand, its reliability will vary between experts, and because its assessments are inevitably somewhat subjective, its reports are likely to be incomplete, biased, and difficult to validate (9). Detailed instructions in the evaluation guide (e.g., detailed process description, clear notation, structure of the statement) can help reduce the subjectivity of this method. A comparison between expert judgment and user observation, and the superiority of the expert judgment are shown in Ref. 10.

We now have a number of guideline-oriented checklists for experts; for example, the checklist of the Bavarian testing authority (11), or the extensive compilation of questionnaires for evaluating the use of new technologies in a company (12). An important measure for guideline-oriented evaluation methods is the extent to which they are embedded in a test scheme (e.g., in a test specification for the performance of an evaluation). Many of them allow the evaluator to choose the way the system being tested should be used to obtain answers to test questions. Along with the test questions proper, some guideline-oriented methods also specify the evaluation procedure (e.g., EVADIS described in the section Example of a Comprehensive and Practical Evaluation Method).

Experimental Evaluation Methods

Among experimental evaluation procedures, “benchmark tests” play an important role. These involve comparing the way different systems perform certain standardized tasks. A case in point is the study by Roberts and Moran (13) involving nine text editors. Benchmark tests do not yield absolute statements about systems but involve placing different systems on an ordered scale based on defined criteria. The comparative nature of benchmark tests does not necessarily apply to other experiments (e.g., experiments testing theories). Well-known examples in this respect are experiments testing the GOMS model (14).

One problem involved in planning experiments is the correct definition of dependent and independent variables; a second problem is the selection of the proper environment for the study. A third problem is the lack of any underlying theory dealing with man-machine interaction, so that the features to be considered are mostly left to the researcher’s imagination and sympathies.

Practical Evaluation: Need of Combining Evaluation Methods

Considering the requirements of a comprehensive evaluation (see the section Context of Use) and the above classifications of methods, we have analyzed different types of available evaluation methods (6,7,11,12,15-23). The result was that there is no single “best” evaluation method (24). All of the examined methods have some disadvantages or consider only a limited number of elements of the context of use. Many of them contain useful ideas or are very appropriate for the evaluation of a specific element of the context of use. What is needed is a combination of different evaluation methods for the different elements of the context of use (25,26). Figure
2 exemplifies a combination of methods. For each element of the overall system (context of use), a specific method has been chosen. No method to evaluate the hardware has been mentioned because there are many useful guidelines and checklists for this purpose (e.g., Ref. 27).

To explore the characteristics of the typical users of the software, interview techniques can be used. It is a good idea to use a standardized questionnaire to reduce the amount of time (subjective method/user report). The tasks that are typically supported by the software and the user’s working conditions can be examined and evaluated using task analysis methods (objective method/observational method). Well-known task analysis methods such as VERA/B (28), KABA (29), or TBS-GA (30) for office work are available, but they all are very expensive and need comprehensive knowledge of ergonomics. Evaluating the user interface, it will be sufficient to use a simplified task analysis in form of a simple guideline [e.g., VBBA (31)]. To assess the usability of the user interface of the software, an expert judgment is very useful (expert method/specialist report). The main advantages are that the expert judgment is relatively fast, uses few resources, provides an integrated view, and can address a wide range of behavior.

INTEGRATION OF EVALUATION METHODS IN THE SOFTWARE LIFE CYCLE

The software development process is typically structured in phases that specify the typical activities of the designer during the development process. There are a large number of different software development life cycle models. All life-cycle models are divided into phases (e.g., Ref. 32).

<table>
<thead>
<tr>
<th>methods</th>
<th>interview techniques</th>
<th>simplified task analysis</th>
<th>expert judgement</th>
</tr>
</thead>
<tbody>
<tr>
<td>elements of the context</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>user</td>
<td>aim: to explore user characteristics tool: standardized questionnaire</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tasks and organisation</td>
<td></td>
<td>aim: to examine and evaluate the tasks and working conditions; to construct a test task tool: guideline</td>
<td></td>
</tr>
<tr>
<td>software</td>
<td></td>
<td></td>
<td>aim: to evaluate the usability of the user interface tool: evaluation software, test items</td>
</tr>
</tbody>
</table>

FIGURE 2 Combination of methods for a comprehensive evaluation.
User Interface Evaluation

- Planning
- Analysis
- Design
- Construction
- Installation and test
- Operation and maintenance

The goal of dividing the development process into well-defined phases is to try to ensure better project planning and permanent control of progress in development. The sequential structure of this life-cycle model has often been criticized, especially concerning usability issues. Iterative or evolutionary approaches have been proposed as alternatives. These often rely on software prototyping, which tries to integrate changing requirements due to user feedback into the development process.

An important point is the timing of an evaluation in the development process of a software system. Timing affects development costs because the costs of design modification are higher during later stages of development. It is, therefore, clear that the evaluation should be an integral part of the whole development process from the beginning. Many of the existing evaluation methods are very difficult to apply in an actual development process, because the costs of applying them seem too high. So system designers need criteria for understanding which evaluation methods are available and useful at different stages of the development process.

Kishi and Kinoe (26) present four criteria:

- The time an evaluation method can be conducted varies because some evaluation methods need real computers or users, whereas others can be used with representational computers or users.
- The type and number of usability problems that the evaluation method can detect depends on the class of usability problems the method is designed to address (e.g., hardware versus software, spatial design versus temporal design, application-specific versus generic).
- The work load imposed by an evaluation depends on the time, the number of people required, the knowledge necessary, and so on, and varies in accordance with the method used.
- Variations in measurement caused by the evaluators are not welcome if a design decision must be based on reliable data (e.g., methods which rely on subjective judgment of evaluators or users).

In reality, no existing evaluation method satisfies all these criteria simultaneously because there are differences between the nature of a development process and the usability evaluation process. Development is more a top-down process, which goes through various stages from functional specifications to implementation, whereas a usability evaluation is more of a bottom-up process. This means that something has to exist before one can use it in a real context and then evaluate it. So, in practice, one needs a combination of different evaluation methods, which complement each other and can be used at appropriate stages of the development process. To reduce the gap between the nature of the development and that of the evaluation process, early or rapid prototyping is a very useful system development method. The prototyping approach should be combined with an evolutionary development process.
This allows the consideration of the results of the evaluation in the development. The design and evaluation processes have to be integrated into the system development process (20).

The proposed combination of evaluation methods (see the section Survey of Existing Evaluation Methods) could be integrated into any life-cycle model of the development process. The exploration of the user characteristics and the examination of the tasks and the organization can be done during the specification of system requirements in the analysis phase. It is one of the aims of this phase to analyze the users and their tasks, to get the necessary information for the evaluation as a by-product of the development process. If a prototyping approach is used, the assessment of the usability of the user interface can be done after the development of the first prototype in the phase design. The results can be taken into consideration during the evolutionary development of further prototypes. If no prototyping approach is used, the assessment of the user interface could be placed in the installation and test phase as part of the final quality control of the software.

For the evaluation of standard software, the life-cycle model again has planning and analysis phases, which are needed to specify the necessary requirements and to plan the introduction process, but no design and construction phases. Instead, these are replaced by a phase entitled selection of standard software. The assessment of the user interface with the help of a evaluation method could be one important aspect of the decision-making process for choosing standard software.

EXAMPLE OF A COMPREHENSIVE AND PRACTICAL EVALUATION METHOD

Based on the assumptions described in the previous sections, the necessary requirements for the development of a comprehensive and practical evaluation approach could be defined. Comprehensive means that the evaluation approach should consider the context of use. On the other hand, the evaluation approach should be practical for user interface designers so that they could use the method during the software development life cycle. Therefore, the approach should be based on a pragmatic combination of evaluation methods: interviews, simplified task analysis, and expert judgment (see the subsection Practical Evaluation: Need of Combining Evaluation Methods). With this pragmatic combination of methods, all aspects of a comprehensive evaluation cannot be covered in all details. The limitations of the approach are described later in the subsection Highlights and Limitations of EVADIS. This new evaluation approach called Evaluation of Dialogue Systems (EVADIS was developed from 1988 through 1992 at the Human-Computer Interaction Research Division of the German National Research Centre for Computer Science (GMD), in close cooperation with the Institute for Applied Computer Science and Information Systems of the University of Vienna. The final version of the evaluation approach EVADIS was published in Ref. 34. The evaluation procedure and its main components will be described in the following sections.

Typical Users of EVADIS

Typical users of EVADIS might be developers and vendors of office software, organizations wanting to buy office software, management consultants, or trade
User Interface Evaluation

unions. Because of its newness, we have no feedback from such organizations about their experiences with the practical use of EVADIS.

Another important area of use for EVADIS is the field of education. Many universities (e.g., University of Vienna, University of Koblenz, University of Dresden, University of Berlin) use EVADIS as a means of instruction for students of computer science. All of these universities have had good experiences with the use of EVADIS in practical courses. Their students are able to learn the basic concepts of human factors and the use of evaluation methods in an enjoyable way. Therefore, EVADIS has been recommended as a good example of an evaluation approach in the “German Curricula for Human–Computer Interaction” (called “Software-Ergonomie”) for computer science at the universities (35).

The Evaluation Procedure and Components of EVADIS

Figure 3 gives an overview of the evaluation procedure and the EVADIS components necessary to execute the five evaluation steps. The EVADIS evaluation procedure consists of the following five steps, which are described in detail in the EVADIS evaluation guide:

Step 1 Installation and exploration of the software to be tested
Step 2 Examination and evaluation of the tasks; selection of relevant test items; construction of test task(s)
Step 3 Exploration of user characteristics
Step 4 Evaluation of the software using the test task(s)
Step 5 Interpretation of the results and drawing up of a test report

The first three steps can be executed simultaneously. The result of these three steps is a test task that is used as a “script” to evaluate the software, and a ranked list of the ergonomic criteria. The order of this list reflects the importance of each ergonomic criterion for the particular user group. The intention is to ensure that the user characteristics are taken into consideration during the evaluation of the user interface as well as during the interpretation of the test results. Step 4 is the central step of the evaluation process. Here, all selected test items have to be answered. They are embedded in the test task. The evaluation process alternates between the test task operations and answering of the associated test questions. The result of these activities is a test record that forms the basis for the interpretation of the results and the writing of the test report in Step 5.

The following EVADIS components are used during the evaluation procedure to instruct the evaluator:

Component A Questionnaire to evaluate the tasks
Component B Guideline for composing test task(s)
Component C Questionnaire to explore user characteristics
Component D List of test items
Component E List of typical functions of office software
Component F Collection of test task examples
Component G Guideline for writing the test report

The description of the evaluation procedure and components of EVADIS that follows is meant to familiarize the reader with the underlying ideas of this evaluation approach.
FIGURE 3 Evaluation procedure and components of EVADIS.
Installation and Exploration of the Software to Be Tested

The first step for the evaluator is to install the software product and to explore the main product characteristics (e.g., product identification, product description, main application areas, major functions). The installation is important, because the evaluator should be able to judge the complexity of the installation procedure. After this step of the evaluation procedure, the evaluator should be familiar with the software product and the necessary hardware requirements (e.g., operating system, input/output devices).

Examination and Evaluation of the Tasks; Selection of Relevant Test Items; Construction of Test Tasks

The aim of the second step is to familiarize the evaluator with the tasks for which the software will be used as an aid. Based on this information, the evaluator can compose test tasks. For this purpose, the evaluator can use four components of EVADIS:

- Questionnaire to evaluate the tasks (component A)
- Guideline for composing test tasks (component B)
- List of test items (component D)
- Collection of test task examples (component F)

The questionnaire to evaluate the tasks (component A) consists of 25 items the evaluator has to answer during the examination of the tasks and working conditions. The questionnaire examines the characteristics of the existing tasks. The questionnaire is based on the suggested list of good task characteristics from ISO 9241 Part 2 (36).

- Do the tasks need a variety of skills appropriate to the user's abilities?
- Are the tasks identifiable as whole units of work?
- Do the tasks provide the user with an appropriate degree of autonomy?
- Do the tasks provide the user with feedback on his performance?
- Do the tasks make a significant contribution to the total function of the system?
- Do the tasks provide opportunities for the development of existing skills and the acquisition of new skills?

The following example shows a typical item for evaluating the quality of the tasks:

<table>
<thead>
<tr>
<th>Component 421: Distribution of work</th>
<th>Test item No.: 421.05.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Criteria 05: Autonomy</td>
<td></td>
</tr>
<tr>
<td>Question:</td>
<td>Can the employee decide in which manner, in which order, and with which methods and tools he or she can accomplish his or her tasks?</td>
</tr>
<tr>
<td>Reference: Bonitz 1989</td>
<td></td>
</tr>
<tr>
<td>Answer options: ( ) no ( ) sometimes ( ) yes</td>
<td></td>
</tr>
<tr>
<td>Notes:</td>
<td></td>
</tr>
</tbody>
</table>

The interpretation of the results is easy because the answer options to the test items are limited. A simple rating form helps the evaluator to write the final statement
about the ergonomic quality of the existing task. This statement will be included in the final test report (Step 5). With the help of such a short questionnaire it is impossible to detect all ergonomic limitations and deficiencies of the tasks at the workplace. That is not the aim of this questionnaire. It is only intended to give the evaluator a first impression of the ergonomic quality of the work and to show him serious deficiencies. If such deficiencies are detected, complete task analysis methods, like VERA/B (28), KABA (29), or TBS-GA (30), should be used for a detailed analysis.

The guideline for composing test tasks (component B) consists of detailed instructions and a collection of query sheets. With the help of the guideline, the evaluator has to examine the tasks that the users of the software are carrying out or plan to carry out. Using a combination of observation and posing questions to the user, the evaluator completes the query sheets. These query sheets include questions on the following topics:

- Description of the working environment and the workplace where the software is used (organizational embedding)
- Overview of the user’s tasks at the workplace and determination of the tasks which are supported or will be supported by the software
- Description of the software-supported tasks and a list of the associated software functions
- Overview of the hardware environment necessary for the software.

Next, before composing the test tasks, the evaluator has to read the complete list of test items (component D), to select the relevant ones and include them in the test tasks. Based on the results of the observation and querying process, the evaluator is able to compose test tasks. The task analysis shows the evaluator which of the software functions are normally used to accomplish typical tasks. It also shows the importance of each function for this purpose. The final test task is a “script” consisting of all functions needed to accomplish one or more typical task(s) and the relevant test items. Depending on their content, the test items are placed after a sequence of test operations. The following example shows a small part of a test task:

```
9. Delete a file
Make a copy of the file “report” and delete the original file.
Test item no.: 241.07.20 Content of item: non reversible action
Test item no.: 320.05.20 Content of item: deletion

10. Print a file
Print the file “report” on the local printer. Specify the layout of the document before printing (format, size, number of copy, etc.).
Test item no.: 120.10.20 Content of item: choice of actions
Test item no.: 320.03.10 Content of item: functionality
Test item no.: 140.06.10 Content of item: output medium
```
User Interface Evaluation

EVADIS includes a collection of test task examples (component F), which guide the evaluator in this composition process. These examples show the typical structure of a test task and how the test items should be embedded in the test tasks.

Exploration of User Characteristics

The aim of this step is to familiarize the evaluator with the main characteristics of the user population. For this purpose, the evaluator can use the questionnaire to explore the user characteristics (component C). It is a collection of questions that the software users have to answer. The questionnaire includes 12 questions about user characteristics like knowledge of and experience with hardware and software. The following example shows a representative question from this questionnaire:

**Question:** Which of the following do you feel you have some knowledge of, possibly from your education, or training courses, or having learned on your own? (You can mark more than one option)

**Answer options:**

1. Basic knowledge
   - how to operate the system
   - about hardware and software
   - about new tasks caused by the use of computers
2. Extended knowledge
   - about programming
   - about operating systems
   - about the design and use of data base systems
   - about human factors

**Notes:**

An interpretation procedure and a classification of typical user groups guide the evaluator during the interpretation of the results. EVADIS distinguishes among four user groups: experienced and regular user, experienced and sporadic user, inexperienced and regular user, and inexperienced and sporadic user (37). Each user group has an associated ranked list of software-ergonomic criteria, which shows the importance of each criterion (high, medium, low) for this user group. The ranked lists are based on a psychological theory called “control concept” (38). These rankings will be used for weighting the results of the evaluation in Step 5. In the final assessment summary – generated by the evaluation software – the criteria are sorted by their weighting. An important criterion should have a higher average rating than a less important criteria. Based on the weighting in comparison with the average rating, the evaluator is able to formulate a differentiated usability assessment.

Evaluation of the Software Using the Test Tasks

The aim of this step is to evaluate the usability of the user interface considering the context of use. In the previous steps of the evaluation procedure, the evaluator
has examined the users, the tasks, and the organizational, technical, and physical environment. Now the evaluator can use the list of test items (component D) consisting of about 150 items to evaluate the various properties of the user interface accomplishing the test tasks. The item list is based on extensive studies of the available literature (especially standards, guidelines, and style guides), on the knowledge and experience of the authors, and on the assessment of existing evaluation procedures. To reduce the number of test items, logically related ergonomic requirements are condensed into one item. The different requirements are presented with the help of a range of answer options. The benefit of such a consolidation is that the handling of the item list is much easier. Nevertheless, the list of test items is only a representative selection of ergonomic requirements. It is up to the evaluator—based on the analysis of the tasks and the user group—to adapt these test items or to create new ones. The evaluation software offers some useful features for this purpose.

All items are embedded in a two-dimensional framework. Figure 4 shows this framework in some detail because it gives an important advantage over other evaluation methods. The first dimension is the technical system components, which distinguishes among four levels of the user interface: the input/output interface, the dialogue interface, the functional interface, and the organizational interface. These are basically inspired by the IFIP model for user interfaces (39). The second dimension is the software-ergonomic criteria. These are primarily based on the dialogue principles proposed by ISO 9241 Part 10 (40) and by DIN 66234 Part 8 (41). The reason for using the ISO principles is their increasing importance for the background of the EC directive. EVADIS includes four further criteria: “availability” of hardware and software, “clearness” of the presentation of information, the influence of the software on “cooperation and communication,” and the mechanism for “data protection.” These four principles have been added because, in our opinion,

![Diagram](image-url)
they represent important ergonomic requirements that are out of the scope of the ISO principles.

Presenting the test items in this two-dimensional framework helps to explain to the evaluator the content of the various items and supports the search for the specific properties being investigated. It also ensures the completeness of the list of EVADIS test items. Figure 4 shows the location of some sample items in the two-dimensional framework:

1. How is the user interface structured?
2. Is it possible to change the dialogue technique in different dialogue situations?
3. Can objects (e.g., documents) from one software module (e.g., word processing system) be copied into another software module (e.g., drawing program)?
4. Is it possible to exchange information with other users using the software?

The following example of a test item shows that each test item includes a question, a set of possible answers, a comment, a field for a rating, a field for the explanation of the rating, and a field for notes.

<table>
<thead>
<tr>
<th>Criteria: Self-descriptiveness</th>
<th>Test item No.: 230.05.40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component 230: Messages</td>
<td></td>
</tr>
<tr>
<td>Group 2: Check at the end of the test task, if necessary with the help of the manual.</td>
<td></td>
</tr>
<tr>
<td>Question: Does the system give status messages on current execution of background processes (processes handled independently of current dialogue)?</td>
<td></td>
</tr>
<tr>
<td>Answer options:</td>
<td></td>
</tr>
<tr>
<td>( ) yes, continuously</td>
<td></td>
</tr>
<tr>
<td>( ) yes, upon request</td>
<td></td>
</tr>
<tr>
<td>( ) can be switched off</td>
<td></td>
</tr>
<tr>
<td>( ) no</td>
<td></td>
</tr>
<tr>
<td>Comment: Status reports are desirable, but the user should be able to switch them off.</td>
<td></td>
</tr>
<tr>
<td>Assessment:</td>
<td></td>
</tr>
<tr>
<td>Rating:</td>
<td>Weighting of the criteria:</td>
</tr>
<tr>
<td>Explanation:</td>
<td></td>
</tr>
<tr>
<td>Notes:</td>
<td></td>
</tr>
</tbody>
</table>

The answer options cover the general spectrum of possible replies given the current state of R&D into software ergonomics. They help the evaluator to describe the ergonomic quality of the software. They are not complete: first, because technical developments or new knowledge may reveal new options that are not included here; second, because they provide only a rough guide for the evaluator in finding
answers and checking functions. Any special feature not considered in the answer options must be noted separately by the evaluator.

Comments have been included with each item to help the evaluator interpret and assess the questions and answers. They are based on the present knowledge of software ergonomics and indicate which of the answer options are currently considered to be the best interface features. Obviously, such comments are subject to rapid change. They are the best indicator of the progress made in software ergonomics and will be regularly updated.

After comparing the analyzed ergonomic quality with the attainable ergonomic quality shown in the comment, the evaluator can provide a rating. This rating is a scaled discrete function that ranges between 1 (full satisfaction) and 5 (nonsatisfaction) and helps the evaluator during the interpretation process of the results. It is important to mention that such a metric evaluation of software-ergonomic quality is very difficult, because the interpretation can vary between evaluators.

In an extra field, the evaluator has the opportunity to explain the rating he has provided. The explanation of the assessment is very important for the interpretation of the results and the drawing up of the test report. Work with the software being evaluated may be interrupted by events that are not addressed in the test items, for example, by system errors (i.e., faulty operations, and faulty or obviously incomprehensible or nonsensical system responses).

Any such occurrence is to be recorded by the evaluator in the field for notes.

Interpretation of the Results and Drawing up of a Test Report
In the final step, the results of the evaluation are combined into the test report. To give the evaluator some advice, a guideline on test report writing (component G) has been integrated into EVADIS. This guideline proposes the structure and the contents for the test report. The structure is based on the suggestions of ISO 9241 Part 11 (5). The test report gives an account of the software-ergonomic properties of the user interface and discusses the answers to and ratings of the test items, including all additional test protocol notes. The evaluator's assessment is mainly based on the ratings and the explanations of the ratings, but it can only be tentative, as sound software-ergonomic findings are not yet available for many interface properties. Interpretation also involves cross-references between different interface properties (e.g., regarding adherence to the principle of internal interface consistency).

The order in which answers are presented will vary from case to case, depending on the purpose. One account of software-ergonomic properties might be arranged by technical components. Such a format would particularly suit the needs of a designer who wants to know as precisely as possible where a trouble spot is located. For other assessments (e.g., involving a decision on whether to buy a software product or not), a criteria-based report may be required.

Table 1 is an example of an assessment summary that is automatically generated by the evaluation software, after the evaluation is finished. The summary shows the average rating for each criterion. The criteria are sorted by their weighting that is determined by the ranked list in Step 1. As useful extra information, the percentage distribution of each rating is shown. For example, an average rating of 3.00 can result from a very good (50% rating 1) and a very bad (50% rating 5)


### User Interface Evaluation

**TABLE 1** Summary of Evaluation Results

<table>
<thead>
<tr>
<th>Criteria No.</th>
<th>Criteria Description</th>
<th>Weighting</th>
<th>Average Rating</th>
<th>Percentage-distribution per rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>Suitability for the task</td>
<td>High</td>
<td>3.5</td>
<td>0.0 0.0 50.0 50.0 0.0</td>
</tr>
<tr>
<td>03</td>
<td>Conformity with user expectation</td>
<td>High</td>
<td>3.6</td>
<td>0.0 0.0 40.0 60.0 0.0</td>
</tr>
<tr>
<td>01</td>
<td>Availability</td>
<td>High</td>
<td>2.5</td>
<td>16.7 33.3 33.3 16.7 0.0</td>
</tr>
<tr>
<td>04</td>
<td>Self-descriptiveness</td>
<td>Medium</td>
<td>1.5</td>
<td>50.0 50.0 0.0 0.0 0.0</td>
</tr>
<tr>
<td>11</td>
<td>Data protection</td>
<td>Medium</td>
<td>2.3</td>
<td>0.0 66.7 33.3 0.0 0.0</td>
</tr>
<tr>
<td>09</td>
<td>Controllability</td>
<td>Medium</td>
<td>3.6</td>
<td>0.0 0.0 40.0 60.0 0.0</td>
</tr>
<tr>
<td>06</td>
<td>Error tolerance</td>
<td>Medium</td>
<td>3.0</td>
<td>50.0 0.0 0.0 0.0 50.0</td>
</tr>
<tr>
<td>08</td>
<td>Suitability for individualization</td>
<td>Low</td>
<td>3.0</td>
<td>0.0 0.0 100.0 0.0 0.0</td>
</tr>
<tr>
<td>10</td>
<td>Cooperation and communication</td>
<td>Low</td>
<td>4.0</td>
<td>0.0 0.0 50.0 0.0 50.0</td>
</tr>
<tr>
<td>03</td>
<td>Clearness</td>
<td>Low</td>
<td>1.0</td>
<td>100.0 0.0 0.0 0.0 0.0</td>
</tr>
<tr>
<td>07</td>
<td>Suitability for learning</td>
<td>Low</td>
<td>1.5</td>
<td>50.0 50.0 0.0 0.0 0.0</td>
</tr>
</tbody>
</table>

assessment. It can also result from a 100% rating 3. So, for the evaluator, it is very useful to see if the average rating is based on an extreme (first case) or on a balanced implementation (second case) of the user interface.

**Evaluation Software of EVADIS**

EVADIS provides computer support for the evaluation procedure. The evaluation software is implemented in Clipper™ and runs under DOS™ on an IBM-compatible PC. For the evaluation, two computers are necessary: one for the software to be evaluated and one for the evaluation software (e.g., a laptop). The evaluation supporting software presents all test items on the screen in the sequence of the test task. The evaluator has to enter the answers, a rating, the explanation of his rating, and perhaps a note. After finishing the test task, the software supports the evaluator in accessing the user interface. It calculates an average mark for each ergonomic criterion and can sort the test items either by technical components or ergonomic criteria. The evaluator is freed of routine work and can concentrate on the activities on the evaluation process.

** Highlights and Limitations of EVADIS**

To describe the highlights and deficiencies of EVADIS, we can use the criteria described in the section Integration of Evaluation Methods in the Software Life Cycle. EVADIS needs a real computer and real users, so the timing of the evaluation in the development process could be after the stage of designing a prototype and having analyzed the tasks and the user characteristics. Therefore, EVADIS cannot
be used during the specification stage of the system development, and the system designer must use a prototyping approach. EVADIS will be primarily used for postevaluation purposes, like evaluating standard software products for purchase decisions. The primary focus of EVADIS is on the software. Therefore, the type and number of problems one can detect are mostly related to software usability, not to the quality of work or to the user's behavior. EVADIS supports expert judgment, so the workload imposed by the evaluation can be restricted. There is also computer support available, which reduces routine work. Much information about the tasks and user characteristics is needed. If it is not specified during the analysis process, the evaluation could be very time-consuming. An expert with a great deal of knowledge about human factors is also needed. Variations in assessment between evaluators are reduced by a detailed evaluation guide, which describes the whole evaluation process. Nevertheless, the final statement can be biased to a certain degree by the judgment of the expert concerning the relevance and rating of the evaluation items. Experimental tests are now available that show the validity and reliability of EVADIS, and an empirical comparison of EVADIS with the evaluation method "ISONORM 9241/10" (42,43) have been made.

SUMMARY

The development of methods and tools for GUI evaluation and design is still an area of ongoing research (44). There is an increasing need for practical and comprehensive evaluation methods and tools for user interfaces. Practical means that the amount of needed time and resources must be manageable in software projects. Comprehensive means that the users, their tasks, the software, and the organizational context have to be considered during the evaluation of user interfaces. The requirements of such a practical and comprehensive evaluation form a framework for the construction or selection of appropriate methods. Only a combination of different evaluation methods is useful to reach these goals. The presented evaluation approach EVADIS is based on such a combination of methods. EVADIS overcomes some typical shortcomings of known evaluation methods. In particular, it takes the task, the user, and the organizational context into consideration and provides computer support for the use of the evaluation procedure. It supports the evaluator during the evaluation process with detailed instructions. It guides the evaluation of the results and allows for several ways of presenting them, depending on the purpose of the evaluation. Typical users of EVADIS are software developers as well as buyers. Also, a number of universities have had good experiences with the use of EVADIS as a means of instruction for students.

REFERENCES

User Interface Evaluation

4. European Economic Community, "The Minimum Safety and Health Requirements for Work with Display Screen Equipment," 90/270/EEC.


43. J. Prümper, "Arbeitsbericht Validierung ISNORM 9241/10/EVADIS II (internal draft), Prümper & Partner, München (1994).


BIBLIOGRAPHY

User Interface Evaluation


HARALD REITERER