An Introduction to Software Usability

N.M. Avouris

University of Patras, ECE Department- HCI Group, GR-26500 Rio Patras, Greece – e-mail: avouris@upatras.gr

Abstract. Interactive computer applications have become ubiquitous and therefore large sections of our societies have become users of computer software. In this context the issue of Software Usability has gained prime importance. So computer professionals and the software industry need to put special attention in this issue. In this paper an introduction to the field of Software Usability is attempted. This is a prominent area of the discipline of Human-Computer Interaction (HCI) that studies interaction of humans and computer machinery and focuses on the design of interactive computer systems. The paper introduces a workshop on Software Usability, organised by the network of Excellence on Software Usability, which has been active for the last two years in Greece.

1 Introduction

It has been observed that today the user interface is often the single most important factor in the success of a software project, as it has been estimated that between approximately 50% and 80% of all source code developed is concerned with it (Myers and Robson 1992). Usability is therefore the software quality factor that has grown in importance during the last years. The field of computer science that provides theoretical background and proposes techniques for producing usable software systems is the Human-Computer Interaction (HCI) field, considered so important by ACM and BCS computing curriculum studies that it is proposed to permeate the entire curriculum of computer science (Faulkner and Culwin 2000).

There are a wide variety of development and evaluation techniques that have been shown to lead to more usable software applications. These techniques, as supported by international standards and accepted practices, shape a new engineering field at the crossroads of software engineering and human-computer interaction, the field of Usability Engineering. This growing interest in the area of Usability is also reflected in the research and professional activities of computer scientists in Cyprus and Greece. Prominent Greek scientists have contributed in recognised breakthroughs in the field through their theoretical and applied work (e.g. Stefanidis, 2000), while the forthcoming Panhellenic Conference in Human-Computer Interaction (Avouris and Fakotakis, 2001) has drawn the interest of

*The reported research is supported by the Network of Excellence on Software Usability funded by the Greek Secretariat for Research and Technology (GSRT).
large part of our research community and software industry. In this context the Network of Excellence on Software Usability (www.ee.upatras.gr/hci/usabilitynet), that has been funded by the Greek Secretariat of Research and Technology during the last two years, has been one of the most active networks in Computer Science.

This paper reviews the state of the art in software usability. An overview of international standards and key terms definitions are included first. Subsequently some techniques for usability evaluation are briefly described and discussed. Finally the perspectives of the field and the implications for software engineering and the industry are outlined.

2. Definition of Software Usability

Software usability (SU) has been the subject of many international standards, directives and theoretical and empirical research during the last years. Since early days it has been considered an important factor of Software Quality (Xenos 2001). Notable documents that are concerned with SU are the ISO Standard 9126 that relates to software quality, ISO 9241 that concerns ergonomic requirements of use of computer equipment and the European Council directive 90/270/EEC on minimum safety and health requirements for work with computer equipment, see (Avouris, 2000 and Fitzpatrick and Higgins, 1998) for discussion of these standards. At the same time many practical techniques for measuring usability have been proposed to be introduced in the interactive software development lifecycle (Nielsen, 1993, Dix et al. 2000, Preece et al., 1995, Schneiderman, 1999 etc.).

Usability was originally related with making systems easy to use and easy to learn, as well as supporting the users during their interaction with computer equipment. There have been however many attempts to relate the term to more attributes and metrics. In ISO 9241-11 draft standard Usability is defined as the "extend to which a product can be used with effectiveness, efficiency and satisfaction in a specified context of use" (ISO 9241). The attributes which a product requires for usability depend on the nature of the user, task and environment. A product has therefore no intrinsic usability, only a capability to be used in a particular context. So usability cannot be assessed by studying a product in isolation.

There are three potential ways in which the usability of a software product could be measured, according to (ISO 9241):

(a) By analysis of the features of the product, required for a particular context of use. Usability could be measured by assessing the product features required in a particular context. Appropriate features are specified in various parts of ISO 9241. However ISO 9241 only gives partial guidance. Of the many potential design solutions compatible with ISO 9241, some will be more usable than others.

(b) By analysis of the process of interaction. Usability could be measured by modelling the interaction between a user carrying out a task with a product. However, current analytic approaches do not give very precise estimates of usability. As the interaction is a dynamic process in the human brain, it cannot be studied directly.

(c) By analysing the effectiveness and efficiency, which results from use of the product in a particular context, and measuring the satisfaction of the users of the product. These are direct measures of the attributes of usability. If a product is more usable in a particular context, usability measures will be better.
Usability defined in terms of the quality of a work system in use necessarily depends on all factors, which can influence the use of a product in the real world, including organisational factors such as working practices and the location or appearance of a product, and individual differences between users including those due to cultural factors and preferences. This broad approach has the advantage that it concentrates on the real purpose of design of a product - that it meets the needs of real users carrying out real tasks in a real technical, physical and organisational environment. This is consistent with the objectives of ISO 9241, described in ISO 9241-1.

According to standard ISO/IEC 9126, usability is an attribute of software quality. According to this standard, the term is used to refer to the capability of a product to be used easily. This corresponds with the definition of usability as a software quality: "a set of attributes of software which bear on the effort needed for use and on the individual assessment of such use by a stated or implied set of users". This is related to the capability of the software product to be understood, learned, used and be attractive to the user, when used under specified conditions. It is observed that there is an inter-relation between some aspects of product functionality, reliability and efficiency that will also affect usability, but for the purposes of ISO/IEC 9126 are not classified as usability.

It is also observed that users may include operators, end users and indirect users who are under the influence of or dependent on the use of the software. Usability should address all of the different user environments that the software may affect, which may include preparation for usage and evaluation of results.

Usability is further analysed in this standard in Understandability, Learnability, Operability, Attractiveness and Compliance. These are briefly described in the following (Bevan, 1997):

Understandability is defined as the capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use. This attribute will depend on the documentation and initial impressions given by the software.
Learnability is the capability of the software product to enable the user to learn its application.

Operability is the capability of the software product to enable the user to operate and control it. Aspects of suitability, changeability, adaptability and installability may affect operability. Also this attribute corresponds to controllability, error tolerance and conformity with user expectations as defined in ISO 9241-10. For a system, which is operated by a user, the combination of functionality, reliability, usability and efficiency can be measured externally by quality in use.

Attractiveness is the capability of the software product to be attractive to the user. This refers to attributes of the software intended to make the software more attractive to the user, such as the use of colour and the nature of the graphical design.

Compliance to standards and guidelines refers to the capability of the software product to adhere to standards, conventions, style guides or regulations relating to usability. In figure 2 the key quality factors according to ISO 9126 are shown.

Figure 2. Usability as an attribute of software quality according to ISO 9126

Additionally to these standards, the European Communities have issued the Directive 90/270/EEC that has been adopted by member states (in Greece this has been the Presidential Decree 398/19.12.1994). According to this Directive the statutory obligations are set, concerning safety and health for work with display screen equipment. The following principles are related to the human-computer interface:

1. The software must be suitable for the task
2. The software should be easy to use and where appropriate, adaptable to the operator’s level of knowledge or experience
3. The system must provide feedback to workers on their performance
4. The system must display information in a format and at a pace which is adapted to operators
5. The principles of software ergonomics must be applied, in particular to human data processing.

Some of the above principles can be found in usability heuristics, discussed in the next section. For detailed instructions on application of this legislation see (Marmaras, 1997).

While in this section more emphasis is provided to standards and legislation relating to software usability, in the next section we proceed with an overview of the current practice of usability engineering, with special emphasis on usability measuring methods.
3. Measuring usability
Many attempts have been reported to further analyse software usability in more practical measurable terms. So for instance in (Nielsen, 1993) usability is analysed in: *Easiness and speed of learning of system use, efficiency to use, easiness to remember system use after certain period of time, reduced number of user errors and easy recovery from them, subjective satisfaction of users*. The emphasis on this proposal is on easiness of measure of usability factors, while some provision is made to accommodate various classes of users, like novice (easiness and speed of learning), occasional users (remember use) and expert users (efficiency of use). Many frameworks have been proposed to measure usability according to these dimensions and evaluate interactive software systems (Nielsen, 1993, Rubin, 1994). Also many attempts have been made to relate these aspects with system performance. For instance, measure of performance can be considered the measure of improved learning, e.g. better understanding by the user of the task and better relation of the task to the available tools and operators. An overview of techniques to measure usability-related factors is included in the following.

3.1 Inspection methods
Usability inspection methods are evaluation methods involving usability experts examining the software user interface. Many inspection methods can be based on specifications that have not necessarily been implemented yet, so they can be performed early in the software lifecycle, though some methods also address issues like the overall system usability concerning the final prototype. The main methods of this category are:

- **Heuristic evaluation** involves usability specialists who judge whether each dialogue element follows established usability principles (the "heuristics"), see also section 3.4.
- **Cognitive walkthrough** uses a detailed procedure to simulate task execution at each step through the dialogue, determining if the simulated user's goals and memory content can be assumed to lead to the next correct action. Variations of this approach have been recently proposed (Demetriadi, et al., 1999).
- **Pluralistic walkthrough** uses group meetings where students, developers, and usability experts step through a learning scenario, discussing each dialogue element.
- **Feature inspection** lists sequences of features used to accomplish typical tasks, checks for long sequences, cumbersome steps, steps that would not be natural for students to try, and steps that require extensive knowledge/experience in order to assess a proposed feature set.
- **Standards inspection**, during which experts inspect the interface for compliance with certain standards. This can involve user interface standards as well as domain-specific software standards, departmental standards if they exist, etc.
- **Guidelines checklist** help ensure that usability principles will be considered in a design. Usually, checklists are used in conjunction with a usability inspection method. The checklist gives the inspectors a basis by which to compare the product.

3.2 Testing methods
These are tests, measuring system performance against pre-defined criteria. These criteria are defined according to the usability attributes, suggested by the usability standards and
empirical metrics discussed in the previous section. Typically we observe individual users performing specific tasks with the system. Data are collected on measured performance. For example, time required to complete the task or number of errors made. Selection of appropriate users and representative tasks is essential. Also a properly designed and organized usability laboratory is important (Avouris 2000, Preece et al., 1995).

The most widely accepted usability testing techniques are: **Thinking Aloud Protocol** is a technique widely used during usability testing. During the course of a test, the participant is asked to vocalize his/her thoughts, feelings, and opinions while interacting with the software, performing a task - part of a user scenario. This technique may be particularly difficult to use with some user groups, like young students, who are distracted by the process, however it provides a valuable insight to user cognitive processes, while interacting with the software. **Co-discovery** is a type of usability testing where a group of users attempt to perform tasks together while being observed, simulating typical work process, where most people have someone else available for help. This can be particularly suitable in many work scenarios. **Performance measurement.** Some usability tests are targeted at determining hard, quantitative data. Most of the time this data is in the form of performance metrics. E.g. required time to execute specific tasks. The ISO 9241 promotes in particular a usability evaluation approach based on measured performance of pre-determined usability metrics. An example of metrics and usability factors is shown in figure 3.

Finally **in-field studies** concern observation of the users performing their tasks in their usual environment of study/work. These techniques have the advantage of the natural user performance and group interaction however they present limitations in terms of measuring performance, since the necessary testing equipment cannot be used in a typical workplace.

<table>
<thead>
<tr>
<th>Usability objective</th>
<th>Effectiveness measures</th>
<th>Efficiency measures</th>
<th>Satisfaction measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall usability</td>
<td>Percentage of goals achieved; Percentage of users successfully completing task; Average accuracy of completed tasks</td>
<td>Time to complete a task; Tasks completed per unit time; Monetary cost of performing the task</td>
<td>Rating scale for satisfaction; Frequency of discretionary use; Frequency of complaints</td>
</tr>
</tbody>
</table>

Figure 3. Examples of measures of usability (from ISO 9241)

### 3.3 Inquiry methods

**Questioners and Interviews based protocols** These techniques prompt the users by asking direct questions about the system. The users’ ability (or lack of) to answer questions can help evaluators decide about parts of the system interface that present difficulties for the users. Many questionnaires have been proposed serving various usability evaluation objectives (Avouris 2000, Schneiderman, 1999).

While inquiry methods can be used to measure various usability attributes, their most common use relates to measurement of user satisfaction. A known technique for measuring user satisfaction is though SUMI, the Software Usability Measurement Inventory, developed by a research group of the University College Cork, to measure user satisfaction, and hence assess user perceived software quality. SUMI is an internationally standardised 50-item questionnaire, available in several languages. It takes a maximum of 10
minutes to complete and needs only small user sample sizes (Kirakowski and Corbett, 1993).
The results that SUMI provides are based on an extensive standardisation database built from data of various software products such as word processors, spreadsheets, CAD packages, communications programs etc. SUMI results have been shown to be reliable, and to discriminate between different kinds of software products. In particular, the SUMI database allows evaluation of a product against what is considered to be the prevailing market norm, and the statistical background to SUMI enables the analyst to pinpoint quite precisely the relative standing of the product being assessed to the market as a whole. SUMI results are analysed into 5 sub-scales: Affect, Efficiency, Helpfulness, Control, and Learnability. These scales have been derived by an iterative process of factor analysis of large databases, and present a view of subjective usability for which there is a high level of empirical support.

3.4 An example: Heuristic evaluation

During heuristic evaluation the interface is carefully examined by a group of system design experts against a list of commonly accepted principles (heuristics). In a typical case of usability evaluation of a multimedia laboratory presentation software, discussed in more detail in (Avouris, Tselios and Tatakas 2001), the experts studied the interface and made their observations in written form. They were also provided with an evaluation sheet in which they could quantify their judgement by assigning marks indicating degree of conformance with each particular rule. The range of assigned values was $[-2 .. +2]$. An importance weight ($w_i$) was also given to each rule, indicating the relevance of the general principle to the system according to the experts' opinion. A quantitative evaluation of the system was obtained by the formula: $e = \sum w_i \cdot r_i$, where $r_i$ the average score of rule $i$ and $w_i$ the relative weight of this rule according to all experts opinions. This approach is particularly useful in case of comparative evaluation of alternative systems.

The heuristic rules of interface design that the experts tested, proposed by (Nielsen, 1993), were the following:

1. Use of simple and natural language
2. Minimization of required memory load from the user
3. Consistency in terminology and symbols used throughout the interface and conformance with the domain norms and standards
4. Degree and quality of system feedback
5. Provision of clearly marked exits and undo
6. Provision of shortcuts for experienced users
7. Informative error messages.
8. Prevention of errors

A group of ten evaluators was engaged in this experiment. This follows empirical studies, which have shown that six to eight evaluators are able to detect about 75% of the usability problems. Heuristic evaluation is performed by letting each individual evaluator inspect the interface alone. This procedure is important in order to ensure independent and unbiased views. The evaluators received instructions to go through the interface at least twice following a typical scenario of use, looking at each element of the interface (for example, each menu item, button, control, etc.) After the evaluation session, a meeting with the
participation of all the experts helped them exchange their views and discuss their individual findings.

Through the overall evaluation graph, areas of problems were clearly shown. The weight factors also helped identify the severity of the problems, so combination of high importance weight and low mark necessitated corrective actions. This view was also accompanied by the written suggestions of the experts. Suggestions were codified and taken into consideration in the next design iteration of the software. In particular suggestions relating to rules of high importance or suggestions with high frequency were given high priority in the list of modifications.

It has been proven that heuristic evaluation can be used at almost any time during the development cycle, although it is probably best suited to earlier stages, when the design is not firm enough. Experts can be provided with paper mock-ups, or even just design specifications, and still get a good amount of usability problems discovered before actual production work begins. However often the experiments are based on late stage prototypes, and are used as overall system quality indices, according to the experts’ judgement.

4 Discussion and Conclusions

An outline of usability engineering definitions and applied usability evaluation techniques have been provided in this paper. One observation should be made at this point that the field of software usability engineering, despite its importance, has not succeeded yet to become a mainstream subject in the field, since there is still a wide gap between the current software engineering practice and the usability engineering technology described in this paper. However, the increased importance of the user in the competitive world in which modern software operates, the increased awareness of our societies in quality-related matters and the widespread use of software products by large sectors of modern societies are rapidly changing this attitude. This change necessitates the active involvement of the usability-related researchers and practitioners. The workshop on software usability, of the 8th Panhellenic Conference on Informatics is another occasion to increase our community’s awareness in this area. The workshop objective has been to draw the attention of the research community and industry in the area of usability engineering, and it is organised as an activity in the frame of the Network of Excellence on Software Usability (www.ee.upatras.gr/hci/usabilitynet), funded by the Greek Secretariat for Research and Technology (GSRT) and co-ordinated by the University of Patras HCI Group.

The main purpose of this network has been to enhance co-operation and sharing of experience and practice of its members in the area of software usability. The Network has been active since the beginning of 2000 and there are activities planned until the end of 2001, while new frameworks are sought, through which to continue this activity. One such framework being the forthcoming Panhellenic Conference on Human-Computer Interaction, which is expected to become a regular event in the future.

1 The partners of the Software Usability Network of Excellence are the University of Patras (coordinator), National Technical University of Athens, Aristotle University of Thessaloniki, University of Ioannina, University of Athens, University of the Aegean, Computer Technology Institute (CTI), Institute of Computer Science (ICS) of the Foundation for Research and Technology, Sunsoft Ltd, Opentec Ltd, DIS-Computer Logic S.A., Knowledge S.A.
Also further research activity and experimentation in the area is urgently needed. New human-computer interaction paradigms necessitate new usability engineering practices. Some active areas of research, also reflected in the SU workshop, are related to formal methods and integration of usability practice in the software analysis and design lifecycle, adaptation of usability methods and techniques to new software paradigms (natural language interfaces (Fakotakis et al. 2001), web and distributed applications (e.g. Tselios et al., 2001), agent-based user interaction (e.g. Avouris and Solomos, 2001) etc.), establishment of usability requirements for specific fields, like educational software, multimedia and virtual reality systems, medical applications and many others.

References

10. Faulkner X. and Culwin F. (2000), Enter the usability engineer: integrating HCI and software engineering, Proc. 5th Annual SIGCSE/SIGCUE Conference on innovation and technology in computer science education, pp. 61-64, July, Helsinki Finland


