Scandinavian Journal of Information Systems

Volume 3 | Issue 1 Article 4

1991

USABILITY AS A GOAL FOR THE DESIGN OF COMPUTER SYSTEMS

Saila Ovaska

University of Tampere, Finland, SailaOvaska@emailaddressnotknown

Scandinavian Journal of Information Systems, Vol. 3, pp. 47–62, 1991

USABILITY AS A GOAL FOR THE DESIGN OF COMPUTER SYSTEMS

SAILA OVASKA
Department of Computer Science at University of Tampere
P.O. Box 607, SF-33101 Tampere, Finland

Abstract

The goal of all software engineering is to construct computer systems that people find usable and will use. Usability is an overall goal that encompasses both system functionality and user interface issues. Assessing usability is vital for those acquiring software packages as well as for those designing and developing software. The concept is also worth scientific research. Still, defining or measuring usability is problematic both in the course of system development projects and in research settings. The measures promoted by some recent usability studies are inadequate and even give rise to false assumptions.

The concept of usability is a difficult one since the factors affecting it are defined only in the use context. It is not possible to evaluate the usability of a computer system without tying it up with the actual activities the user wants to use the system for. This claim is supported by a series of case studies in decision support systems. These studies have clearly shown that no list of usability evaluation criteria suffices in the long run; instead, perceived usability depends heavily on (organizational) context. Still, it is an important goal in every context, not only in decision support systems.

Keywords: Usability assessment, contextual research.

1 Introduction

Why are interactive computer systems (or computerized information systems)[1] sometimes not used by people who might benefit from them (cf. Nickerson 1981, Nielsen *et al.* 1986)? Is it just because of design errors in the user interface? And why are people sometimes willing to use poorly designed systems with clumsy user interfaces? Is it because they have never experienced better ways to do things? Obviously there are *usability problems* that could be taken care of with better design and development efforts — or with more systematic usability evaluations before the acquisition of new software.

Some of the recent usability studies need to be criticized, because the user interface is designed and evaluated per se, without taking into account the real users and the tasks they are trying to accomplish with the system. Only some of the user interface researchers, e.g. Whiteside *et al.* (1988) and Bødker (1987), point out the importance of the use context in defining and evaluating usability. Evaluating a user interface without taking the user and the goals of the user into consideration may focus attention towards such details of the system that the user would not find problematic. As Bødker states it, the user interface cannot be seen independently of the goal of the user, or of the other conditions of the use activity (Bødker 1987, p. 147).[2] Trivial as it may seem, the relationship shown in Figure 1 is essential when the usability of a computer system is to be evaluated.

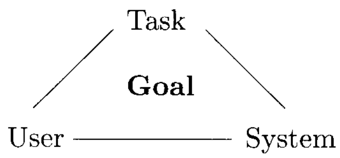

Figure 1: *Usability relationship (Bennett 1983)*

The usability of a computer system is the totality of all *factors that enhance or make possible the user's goal attainment* while using the system in a certain task or in a certain task domain, if the computer system is usable in a set of interrelated tasks. As Goodwin (1987) notes, usability relates tasks to people. Some users may find a system usable while other users don't, though their tasks resemble each other.

Proper functionality is tightly connected with usability, since if the system does not offer the necessary functions, it will remain unusable. Still, increasing functionality is only a partial remedy. Also the way in which users actually access the functions matters, as indicated by studies with different user interfaces (Goodwin 1987, Norman 1986).

Usability issues are vital in every context of use, regardless of the field of application. They are important in situations where users have little influence on the way work is organized. In these situations a usable computer system may enhance work productivity and pleasure, reduce error rates or even produce measurable cost savings (Goodwin 1987). In situations where users may choose the way the work should be done, as is the case with some managers, usability of the computer system is as important. If the system is of little use it may be left unused or delegated to others.

In this paper, managers and their working activities are taken as an example. The field of decision support is chosen for study for two reasons: the managers needing decision support may design their own work and the way they utilize support systems, and they must assess by themselves the usability of the support system before it may become an effective tool in their work. With tools, usability issues relate to the level of specialization (Norman 1986). As in the case studies reported in Section 4, a tool with many functions may result in a situation where only a minor part of the functions is used, but the managers are unable to hide away the unwanted menu choices, and to correct the functionality to better suit their needs.

The goals of supporting managers in managerial decision making (Keen & Morton 1978) include increasing the *effectiveness* of decision makers in solving semi-structured problems. Vaguely stated as the goals of support are, developing a decision support system (DSS) for an individual or a group is always a contextual matter. For managerial use of a DSS, user interface issues are vital, since the use is voluntary. Especially ease of learning seems to be essential: managers have only limited time to get to know the system, and if that time is insufficient to convince them of the pros of the system, it will remain unused. The development of a DSS is, in addition, an evolutionary and iterative effort. For these reasons usability is proposed as the main goal of DSS development.

Usability is even more important than effectiveness, at least when it comes to senior management. Instead of enhancing managerial effectiveness with analytical tools for rational decision making, the design efforts of support systems should be aimed at providing the managers with tools they find *useful* in their tasks, perhaps correcting the misdirected functionality (Winograd & Flores 1986). This shift in perspective brings the support system close to the actual manager and his or her working tasks, as a personal tool. Instead of looking at effectiveness as a goal defined by the company, the manager becomes responsible for effectiveness in his or her work. Computer systems are merely helping in that task, not fostering rational behaviour.

The goals of usability and effectiveness are interconnected, though. If a manager finds a support system usable, it will enhance managerial effectiveness, or at least efficiency (Keen & Morton 1978). Still, managers do not have methods for evaluating usability, and systems development projects seem to be guided by organizational and technical details instead of usability concerns (Gould *et al.* 1991, Burnes 1991).

Defining the factors that affect usability and evaluating usability is a task encountered in every system development process where the users themselves are not capable of developing their own systems. Therefore, a short survey on different methods for assessing usability during system development is presented in Section 2. The methods are partly applicable also when the focus is on usability research, as presented in Section 3. But some recent usability studies point out also the need for deeper analysis of usability in real use context. Some practical evidence from case studies is reported in Section 4, showing not only the difficulty of assessing usability but also a satisfying behaviour in enhancing it.

2 Usability Engineering Methods During a Software Development Project

Usability engineering has become an integral part of software engineering, and some of its methods (Gould & Lewis 1985) are widely accepted:

- early focus on users and tasks,
- empirical measurement, and
- iterative design.

All aspects of usability, including not only user interface but also the help system, training plan and documentation should be developed in parallel (Gould *et al.* 1991).

Most of this knowledge on usability relates to the usability engineering process, and to general guidelines of good interface or product design (Jørgensen 1989). The latest collection of *general usability guidelines* contains nearly one thousand rules, and they are not very useful in normal usability engineering practices (Nielsen & Molich 1990). Merely finding the relevant guidelines from the bulk is a lengthy process, and the results are often not considered worth the effort (Molich & Nielsen 1990).

Also *practical guidelines* for usability engineering exist (Nielsen 1989, Ravden & Johnson 1989, Jørgensen 1989, Nielsen & Molich 1990, Chapanis & Budurka 1990). These guidelines are well accepted, but still not so widely used (Gould *et al.* 1991). In most cases they are incorporated into the design process so late that their impact on the design is minimal. A particular guideline may even be interpreted differently by different designers lacking experience in human factors, and some of these interpretations may result in systems that are not easy to use (Chapanis & Budurka 1990).

How should a systems developer take care of setting and meeting the usability goals? There is clearly a need for practical usability engineering methods that would help the developers to capture the usability problems and their causes. Practical evidence shows that only a small percentage (20–50%) of the possible usability problems are encountered by developers (Nielsen & Molich 1990). Even the most experienced usability engineers do not catch all the usability problems when evaluating an interface. Generally, the possibility of recognizing the problems increases when the evaluation is done by several independent evaluators.

When a systems development project is started, the first task is to decide who the users will be (Gould 1988).[3]Users are not passive organisms but active agents: it is a responsibility of the users to inform the system developers of their requirements and needs (Bannon 1989). When designing a software product, it may be difficult to find potential users of the product at an early stage of development, but tests involving real users working with a full system can capture some otherwise unpredictable usability problems (Marshall *et al.* 1990).

It is important to set behavioural criteria — e.g. ease of use — for the system at the outset of a design project, put these criteria into specific, testable terms, and review and test against these specifications as development proceeds. This design practice is the same, regardless of the system being built, but the specific behavioural criteria and thus the aspects of usability vary (Gould *et al.* 1987). Not all aspects of usability can be defined in measurable terms, though (Whiteside *et al.* 1988). The testing methods may vary from informal to formal evaluations (Whitefield *et al.* 1991).

Most developers use some sort of *heuristic evaluation* of user interfaces instead of turning to the collections of good user interface guidelines (Nielsen and Molich 1990). This means that the developer has an idea of what a good interface looks like. By looking at the interface and walking through or testing the different paths of the program the designer tries to come up with an evaluation of what is good and what is bad about the interface. This method is called *specialist reports* in (Whitefield *et al.* 1991). The specialist bases the evaluation on an image of the user, but still, the evaluation is dependent on the evaluator.

Measuring usability more accurately than with such heuristics is often considered problematic or even impossible. Several usability measures are being promoted for evaluating user interfaces (Whiteside *et al.* 1988). These metrics are meant to be used as engineering tools, and they usually involve user interface *testing in a laboratory setting* with some representative users and tasks defined by the system developers. Usually these standard measures for usability concentrate on observable manifestations as ease of learning, ease of use, throughput, productivity and preference. Still, there is a danger in applying these standard usability measures that we may be evaluating the wrong aspects of the system. As usability is defined from the developer's point of view, these measures might not capture those features of the system that mean the most to the users.

Ravden and Johnson (1989) describe a practical usability evaluation method that uses an *evaluation checklist*. The method is intended for use by the development team together with the users (or evaluators that may not be the actual users for whom the system is designed). The evaluation is started by users who work on a task designed by the developers, and a general checklist is completed by the users after they have finished the task. The checklist contains questions on several aspects of usability. The questions are based on general user interface guidelines (see Table 1). They are divided into 9 sections, each from 13 to 17 questions, and the user is expected to answer questions like "Does the system validate user inputs before processing, wherever possible?", or "Where interface

Visual clarity	Information on the screen should be clear, well-organized, unambiguous and easy to read.
Consistency	The way the system looks and works should be consistent at all times.
Compatibility	The way the system looks and works should be compatible with user conventions and expectations.
Informative feedback	Users should be given clear, informative feedback on where they are in the system, what actions they have taken, whether these actions have been successful and what actions should be taken next.
Explicitness	The way the system works and is structured should be clear to the user.
Appropriate functionality	The system should meet the needs and requirements of users when carrying out tasks.
Flexibility and control	The interface should be sufficiently flexible in structure, in the way information is presented and in terms of what the user can do, to suit the needs and requirements of all users, and to allow them to feel in control of the system.
Error prevention and correction	The system should be designed to minimize the possibility of user error, with inbuilt facilities for detecting and handling those which do occur; users should be able to check their inputs and to correct errors, or potential error situations before the input is processed.
User guidance and support	Informative, easy-to-use and relevant guidance and support should be provided, both on the computer (via an on-line help facility) and in hard-copy document form, to help the user understand and use the system.

Table 1: *Usability guidelines (Ravden & Johnson 1989)*

metaphors are used, are they relevant to the tasks carried out with the system?", on a 4-point scale ranging from *always* to *never*. In addition, the checklist has also two sections of questions on general usability problems encountered during use.

This sort of checklist is quite difficult — and time-consuming — for the users to fill in. The terminology used is that of the developers'. The user would never think about "consistency" or "compatibility" but would probably be lost in the system and puzzled about its use, if it did not behave according to expectations. There is also another problem connected with these questionnaires: the user may find parts of the software usable, the other parts difficult to grasp. As the checklist is aimed at giving an overall evaluation of the software, it might not provide advice for the developers. What corrective actions would have to be taken if the user's answer to a question on software compatibility with user expectations is *some* *of the time*? Still, using this kind of a checklist as a developer tool for heuristic evaluation of user interfaces would probably make it easier for the developer to identify usability problems. It would, however, not help the developer in focusing attention towards the most important aspects of usability.

Learning about the expectations and needs of users — *contextual research* (Whiteside *et al.* 1988) — is critical in usability engineering. The level of success in meeting the user requirements, expectations and needs defines also the usability of the computer system. One should observe actual user behaviour with the present system to learn about the contextual aspects of usability. During development the user must be provided with some sort of a prototype or test system in order for the system developers to get information on the usability of the resulting system (Bannon 1989).

In their human factors evaluation framework, Whitefield *et al.* (1991) mention also situations where the observation of a user using a system prototype is not necessary, but the researcher may rely upon opinions or data reported by a user on some parts of the system under development. The drawback of this method is that the data received may be inaccurate. The only way of getting accurate data is observation of use.

Laboratory experiments, interviewing users or developing questionnaires will not suffice in contextual research. Whiteside *et al.* (1988) point out that their value is minimal. This may be too critically stated: e.g. there may be difficulties in generalizations from the lab environment to the real work context, but still there are situations where lab tests are appropriate and useful. Even though there is an ecological gap (Thomas and Kellogg 1989) between laboratory environments and the real work context, this gap can be bridged. Laboratory experiments need to be tied up with a thorough analysis of the use context and actual observation of use. In this way developers may find out what a user really wants.

If the system is developed by using a software package, there are additional problems in assessing usability. Requirements analysis and field testing are seldom done thoroughly enough to give an adequate picture of the usability of the system, resulting in underutilized systems (cf. Nielsen *et al.* 1986). In general, developing software packages or products imposes even greater responsibilities on the designers — they should make their packages easily tailorable to all possible users.

In the next chapter the view is broadened from usability engineering to *usability research*, formulating theories and models on usability, defining various aspects of usability and finding a measure for usability. As usability is perceived by a user in a specific context, is it at all generalizable, over applications and work activities?

3 Striving Towards Models and Generalizations

Contextual research methods are specifically adapted to one project at a time, and the research findings are difficult to generalize. Still, generalizability is often sought for to help in learning from experiences. It is difficult to define the concept of usability, since the factors affecting usability differ from user to user, and the mutual relationships between the factors are excessive. Some kind of conceptual analysis is, however, needed before any modelling or measurement is possible. No generic metric or questionnaire for assessing all aspects of usability in all cases has so far been developed. Preparing such a measure is immensely difficult if not impossible, and applying it will cause interpretation difficulties and a danger of still omitting some important aspects of usability.

Contextual research is only to some extent accepted as a scientific research method. In contextual research, 'generalizable' usability concepts *emerge* from a series of studies in the same application area, e.g. text processing. Thus the users describe their work with the computer using concepts like 'Control' (Whiteside *et al.* 1988, p. 813). The concept is accepted and used in contextual interviews by other users only if it is one of their key concepts of usability of this kind of an application.

Research studies aiming at *statistical* generalizations and modelling are very common in assessing usability or related properties, e.g. User Information Satisfaction (Bailey & Pearson 1983) and the usability metric provided in (Davis 1989). These studies are typically based on an excessive statistical analysis, and the measures promoted by them are further tested and possibly redesigned in other studies. The models and measures are often heavily criticized for different reasons.

In order to override the difficulty of defining usability, researchers have searched for other more easily definable concepts that represent it partially. By selecting and defining the concept, the researcher also develops a model of the factors that are allowed to be present in analysis. There is a danger of limiting the model too much so that its relevance in real life is minimal (Whiteside & Wixon 1988, p. 357). On the other hand, the more limited a model is, the easier it is to control, and the more scientifically grounded results can be gained. In laboratory experiments there is a need for keeping the situation and all the affecting factors under control. In this way the results gained may be analysed more thoroughly and the causalities found may be reported. It is presumed that if the experiment is done another time, the results would be the same, but in practice this would hardly occur. The generalizability of the findings of a strictly limited and controllable laboratory experiment into the actual tasks of real users is not at all certain, though some researchers, e.g. Mason (1988) have pointed at the opposite.

Davis (1989) claims that usability of a computer system consists of *perceived usefulness* and *perceived ease of use*, among other things. He proposes questionnaires for assessing the user's perceptions, and develops them through pretest (colleagues at MIT), study 1 (with 120 professional users at IBM) and study 2

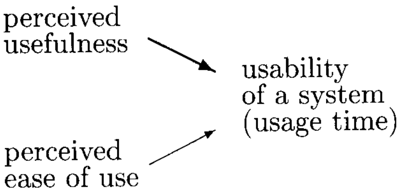

Figure 2: *A model on usability*

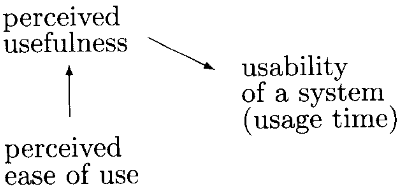

Figure 3: *Another proposition of the factors affecting usability*

(40 student participants in a lab study). Thus the measures have been thoroughly tested, and their internal consistency level is good. The test population is quite technically oriented, though. As a major result, Davis reports that perceived usefulness has a significantly greater correlation with usage behaviour than perceived ease of use has (see Figure 2, where the greater correlation is denoted by a thicker line). In the study, ease of learning is regarded as one substratum of the ease of use construct, not as a distinct construct, and all the other factors affecting usability are left out.

Davis uses usage, i.e. the time user spends with the system, as a surrogate for usability. The more time a user spends with a system, the more usable he or she finds it. There are several flaws in this way of thinking. For instance, the system may be so complicated that the user has difficulties in getting any results and spends consequently time in vain, or the system usage may be obligatory in some sense. Actually, even the usage time is somewhat vaguely defined in the study, since its measuring is up to those who respond to the questionnaires. As a possible correction to the model in Figure 2, Davis suggests that according to the regression data, ease of use may actually be a causal antecedent of perceived usefulness (see Figure 3), and not a direct determinant of system usage.

The research setting described above poses also another question. How could a user gain any perception of a computer system without first using it? In this case, the causalities of Figure 2 would be completely switched (see Figure 4). If a motivated user tries to learn a computer system, initial difficulties encountered in the system will not hinder him or her from utilizing it.

A partial answer to the rhetoric question posed above is presented in (Nielsen *et al.* 1986): a user views a system *approachable* or *unapproachable*, depending on the amount of learning needed before the system is usable. All the factors affecting approachability are not known, but e.g. the size of the manual matters.

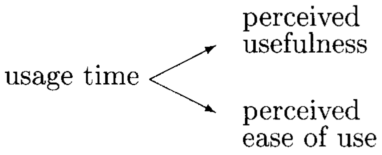

Figure 4: *Rhetoric proposition for a usability model*

Clearly, approachability would have to be included in the analysis of factors affecting perceived usability.

It is hard to learn anything actually new or surprising from (Davis 1989). Its practical value in usability engineering as well as its theoretical value in usability research can be questioned. The initial concepts are not taken from the actual use context, the model is kept simple and the questionnaires generally formulated. The statistical analysis is excellent, though.

Another longitudinal, laboratory-based research study on usability is presented in (Kirakowski & Corbett 1988). Kirakowski and Corbett propose a system independent evaluation metric questionnaire for measuring *user satisfaction* and thus the usability of the system. Their CUSI (Computer User Satisfaction Inventory) metric consists of questions addressing users' feelings of mastery over the computer system (*competence subscale*) and users' feelings of fear or pleasure (*affect subscale*). Instead of attaching the actual CUSI questionnaire to the paper, they show nicely growing competence and affect figures that are based on longitudinal results of changes in users' feelings in ten separate tasks. These results are then compared with other sources of information, and the CUSI metric is thus validated — it measures what it is supposed to measure. The tests are done in a laboratory environment with small scale tasks prepared by the researchers.

Usability is much more than user friendliness or user competence and affect. Measures like CUSI are important, but the conceptual background and the actual questions asked are more interesting than mere application of the measure. These would be needed if someone would like to reapply the measure in some other test, or to conceptually analyse it. The applicability of CUSI would also be of interest in users' real work context, since labour intensive research methods are needed also in contextual research. There is, though, a lack of formal research methods to assess the match or mismatch between tasks and software capabilities (Nielsen *et al.* 1986).

According to Knowles (1988), any technique to evaluate usability is itself not very useful if it has to be restricted to a limited type of task environment. Laboratory environments are a very limited type of task environments. Is a laboratory-defined construct applicable in the real world and will it find the most important aspects of system usability? In the following, findings from a contextual research study in the field of decision support systems are reported.

4 Evidence From a Series of DSS Studies

A series of case studies (Ovaska 1990) was conducted in 1989 in four Finnish companies that use some sort of computerized support system in financial top level decision making. The case studies aimed at recognizing the factors affecting the usability of computer systems in the work situations of financial managers. The studies involved interviews and short observation sessions, and the basic research aim was to understand the tasks these managers were accomplishing with their computer systems. As could be expected, all of the interviewed managers used small spreadsheet applications they had usually prepared for themselves. Also electronic mail and bulletin boards were in use in a couple of the companies. The main research interest concentrated on the factors affecting the usability of a Finnish financial planning and analysis tool called Mikro-Trennus, which is an application package mostly sold to banks and insurance companies. The program is menu-based, and it includes the basic, traditional business graphics. It is designed to be applicable in all sort of business areas, and it has broad functionality. When the managers were asked about the reasons of acquiring this particular software package, they reported it was cheaper than its competitors.

The functionality of the package is based on a recommendation given by an official funding board. The recommendation specifies the standard practices of evaluating solidity, liquidity and cash flows. In practice, accounting measures and language seem to differ greatly from company to company, and there exist innumerable variations of these measures. The size and maturity of the company, the accounting and reporting packages used, and individual likes and dislikes seem to have an influence on the adopted practices and the language used.

Contrary to the model presented in Figure 2 on the relationship of perceived ease of use and perceived usefulness, the managers were unwilling to spend their scarce time on learning to use a system. If they considered a system "too difficult" for their use, they just left it unused after some initial trials, even though they had expectations about its possible merits in their work. Thus at least in the work context of these interviewed managers ease of learning and ease of use were considered important.

The managers found the planning and analysis system difficult to grasp: it takes a lot of time before the user knows how to proceed with it, just which keys to use. Two of the five interviewed managers used it about three times a year, so every time they had to bring in mind how to use it. As the application is also an analysis tool, it offers several possible sets of different analysis techniques and practices to be adopted. The financial managers found the generality of the system a burden; there was no way of eliminating the unwanted menu choices or changing the basic logic to get the results.

The basic logic of the planning tool — the accounting language chosen — is not the same as is used in normal accounting practices in the two companies that use the tool regularly. It causes the managers to rethink the concepts over and over again when using the program. Some of the predefined accounting measures (e.g. Return on Investment) are calculated slightly differently in the application package. One financial manager told that he always had to redo the calculations manually to find the differences between the calculations presented by the tool and the calculations required by top management.

This study indicates that no set of predefined accounting functions in a planning model suffices from company to company, or from year to year. The user needs an easy way to define his or her own functions, and drop out those that are not needed. Some of the changes needed are on the user interface level, but some of them require changes also to the functionality of the system. Thus the usability of the planning and analysis application program depends not only on its user interface, but also on the accounting principles that are used in defining its functionality. A financial manager complained of its suitability for their company: it seems to be a powerful tool but they ought to change all their internal accounting practices to make use of it. After initial trials they had decided not to use it.

It was interesting to note that the managers could have contacted the supplier of the package who was willing to update the system for them — for some extra cost, of course. But the managers kept on using the version they had got, and when a new version with minor improvements was sent to them, they were too busy to install it, though the amount of work required was kept to the minimum. They had no assistance in maintaining the system, and they had no time to learn how to install the system and how to use the updated version. Once they knew the old, cumbersome system, it was enough for them at least for the time being. This kind of satisfying behaviour is reported in (Nielsen *et al.* 1986). It points out the difficulty of assessing usability. The usability of the package would be enhanced if the system were updated. But if the gains of the extra work are not big enough for the managers, they'll be satisfied with the current version.

The negative reactions about the usability of the planning and analysis tool raise a question: why do some of the managers still keep on using it? The reason is simple. They find it useful in their work, and it gives them new ideas for planning. It helps them to see the effect of certain possible outcomes. They do not have a better choice if they want to get any results at all. They are highly motivated in using it. But should the program as such be valued as a usable tool? It shouldn't: correcting some of the design errors would certainly enhance work productivity and make the tool more enjoyable to use. Still, how should one proceed in analysing the ticks these managers would mark on a questionnaire asking if this program is useful in their work, rated on a 7-point scale? Merely filling up a questionnaire would not have given so rich data for the researcher.

How to evaluate usability, when the user is a manager, using the system in work activities? The methods of contextual research should be used, possibly accompanied with other research methods. The manager needs hands-on experience of the system with his or her own data before its pros and cons can be found. Also there is a need of tailoring the system according to these first evaluations; the tailoring should be done by the system supplier before the system is sold, and the system should include facilities for the users to change it easily during the use.

The kind of qualitative evidence reported here is of course not generalizable, but it seems to propose some ideas. For managers, computers are not an end but a means to reach an end. A DSS is brought into an existing environment. It should fit onto the organizational (accounting) practices so that data from other applications can easily be used in the tool. But even more important is the language used: it should be adaptable to that of the users in the organization. There should be no different meanings of the (accounting) terms used. Flexible and customizable user interfaces that are easily learned are looked for. And the users should be provided with a possibility to adjust even the functions defined in the system, not just the user interface.

5 Conclusions

In systems development, usability enhancement is a goal shared by all interested parties: users, systems developers as well as researchers. Enhancing the usability of a computer system will result in an increase in users' work productivity, and make the system an enjoyable tool for the users. While the goal of usability enhancement is commonly agreed upon, all parties lack knowledge of usability. Most of all, labour intensive evaluation methods to be used in systems development projects need to be studied and developed. One problem of the methods is how to capture the richness of the use context, and all the factors that affect usability in a certain use context.

The usability of a computer system must always be evaluated in the real work context to find out user requirements for functionality, difficulties encountered in use, and proper language to be used. Also other contextual factors may affect usability. The importance of observation of use cannot be too loudly stressed as a method for evaluating usability. In addition to observation and interviews also questionnaires and laboratory experiments may be used, if they are tied closely enough to the actual use context. These usability evaluation methods and measures are needed during software engineering projects. At the same time, more systematic usability evaluation is important also in systems development projects where software packages are used.

Managers do not evaluate thoroughly the usability of software packages before acquisition. Instead of usability, other factors seem to have an impact on the acquisition. It seems to be enough that some needed results can be got by using the system; minor deficiencies in the system do not matter so much, and the user grows soon used to them. Still, especially for a manager, usability evaluation before software acquisition is important, and equally important is to require improvements in usability before the acquisition. Poor usability increases the time needed to get the desired results, but it may also increase errors and cause misunderstandings — even misjudged decisions. If the system is not usable, it will remain unused after some initial trials, or its use will be delegated to subordinates or staff. In this case, it is hard to believe that the system would affect managerial work or the decisions made.

It is apparent that all factors affecting usability can not be listed. Two factors emerged in the study of managerial use of a planning tool: the need of tailorability and ease of use. But the most important of all is still proper functionality. If the functionality is sufficient, the manager can overcome the other usability problems if motivated enough. The usability of the planning tool could still be enhanced by reshaping the needed functions and leaving out those not needed, if the system were easily tailorable.

Notes

1. The term computer system is used throughout the paper, as it is shorter, even though "computerized information system" would be more accurate.
2. "Activity" is a term used in the human activity approach, not in contextual research, but in spite of differences, both approaches emphasize that a user interface is not to be designed separately from the use context.
3. Actually there is some controversy on even the concept of a "user" (Bannon 1989). A user is a non-edp specialist, but still not naive nor incompetent in his or her field.

Acknowledgements

Thanks to Matti Jakobsson, Erkki Mäkinen and Matti Pettersson for comments on an earlier version of this paper, and my inspiring working group fellows at the 13th IRIS for encouragement. I wish to acknowledge also the financial support from the Doctoral Programme on Information Systems in Finland. And finally, thanks to the anonymous referees for their clarifying comments and suggestions for new references.

References

Bailey, J. L., & S. W. Pearson, (1983). Development of a Tool for Measuring and Analyzing Computer User Satisfaction. *Management Science* 29(5): 530–545.

Bannon, L. J., (1989). Discovering the Human Actors in Human Factors. DAIMI PB-290, Aarhus University.

Bennett, J., (1983). Analysis and Design of the User Interface for Decision Support Systems. In J. Bennett, ed., *Building Decision Support Systems.* Addison-Wesley, Reading. 41–64.

Burnes, B., (1991). Managerial Competence and New Technology: Don't Shoot the Piano Player — He's Doing His Best. *Behaviour & Information Technology* 10(2):91–109.

Bødker, S., (1987). Through the Interface — a Human Activity Approach to User Interface Design. Ph.D. Thesis, DAIMI PB-224, Aarhus University.

Chapanis, A., & W. J. Budurka, (1990). Specifying Human-computer Interface Requirements. *Behaviour & Information Technology* 9(6): 479–492.

Davis, F. D., (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly* 13(3): 319–340.

Goodwin, N. C. (1987). Functionality and Usability. *Comm. ACM* 30(3): 229–233.

Gould, J., & C. Lewis, (1985). Designing for Usability: Key Principles and What Designers Think. *Comm. ACM* 28(3): 300–311.

Gould, J., S. Boies, S. Levy, J. Richards, & J. Schoonard, (1987). The 1984 Olympic Message System: A Test of Behavioral Principles of System Design. *Comm. ACM* 30(9): 758–769.

Gould, J. (1988). How to Design Usable Systems. In M. Helander, ed., *Handbook of Human-Computer Interaction.* Elsevier Science Publishers B. V., Amsterdam. 757–789.

Gould, J. D., S. J. Boies, & C. Lewis, (1991). Making Usable, Useful, Productivity-enhancing Computer Applications. *Comm. ACM* 34(1): 75–85.

Jørgensen, A., (1989). Using the Thinking-Aloud Method in System Development. In G. Salvendy, & M. Smith, eds., *Designing and Using Human-Computer Interfaces and Knowledge Based Systems.* Elsevier Science Publishers B. V., Amsterdam. 743–750.

Keen, P., & M. Scott Morton, (1978). *Decision Support Systems: An Organizational Perspective.* Addison-Wesley, Reading.

Kirakowski, J., & M. Corbett, (1988). Measuring User Satisfaction. In D. M. Jones, and R. Winder, eds., *People and Computers IV.* Cambridge University Press, Cambridge. 329–338.

Knowles, C., (1988). Can Cognitive Complexity Theory (CCT) Produce an Adequate Measure of System Usability? In D. M. Jones, & R. Winder, eds., *People and Computers IV.* Cambridge University Press, Cambridge. 291–307.

Marshall, C., B. Mcmanus, & A. Prail, (1990). Usability of Product X — Lessons from a Real Product. *Behaviour & Information Technology* 9(3):243–253.

Mason, R., (1988). Experimentation and Knowledge: A Pragmatic Perspective. *Knowledge: Creation, Diffusion, Utilization* 10(1): 3–24.

Molich, R., & J. Nielsen, (1990). Improving a Human-Computer Dialogue. *Comm. ACM* 33(3): 338–348.

Nickerson, R., (1981). Why interactive computer systems are sometimes not used by people who might benefit from them. *Int. Journal of Man-Machine Studies* 15, 469–483.

Nielsen, J., (1989). Usability Engineering at a Discount. In G. Salvendy, and M. Smith, eds., *Designing and Using Human-Computer Interfaces and Knowledge Based Systems.* Elsevier Science Publishers B. V., Amsterdam. 394–401.

Nielsen, J., R. L. Mack, K. H. Bergendorff, & N. L. Grischkowsky, (1986). Integrated Software Usage in the Professional Work Environment: Evidence from Questionnaires and Interviews. In CHI'86 proceedings, *SIGCHI Bulletin*, Special issue, 162–167.

Nielsen, J., & R. Molich, (1990). Heuristic Evaluation of User Interfaces. In CHI'90 proceedings, *SIGCHI Bulletin*, Special issue, 249–256.

Norman, D. A., (1986). Cognitive Engineering. In D. A. Norman, & S. W. Draper, eds., *User Centered System Design: New Perspectives on Human-Computer Interaction.* Lawrence Erlbaum Associates, Hillsdale, New Jersey. 31–65.

Ovaska, S., (1990). Päätöstukijärjestelmän käytettävyyden arviointi. University of Tampere, Dept. of Computer Science, Report B-1990-5. In Finnish.

Ravden, S., & G. Johnson, (1989). *Evaluating Usability of Human-Computer Interfaces: a Practical Method.* Ellis Horwood, Chichester.

Thomas, J. C., & W. A. Kellogg, (1989). Minimizing Ecological Gaps in Interface Design. *IEEE Software* 6(1): 78–86.

Whitefield, A., F. Wilson, & J. Dowell, (1991). A Framework for Human Factors Evaluation. *Behaviour & Information Technology* 10(1):65–79.

Whiteside, J., J. Bennett, & K. Holtzblatt, (1988). Usability Engineering: Experience and Evolution. In M. Helander, ed., *Handbook of Human-Computer Interaction.* Elsevier Science Publishers B. V., Amsterdam. 791–817.

Whiteside, J., & D. Wixon, (1988). Discussion: Improving Human-Computer Interaction — a Quest for Cognitive Science. In J. M. Carroll, ed., *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction.* MIT Press, Cambridge, Massachusetts. 353–365.

Winograd, T., & F. Flores, (1986). *Understanding Computers and Cognition.* Ablex Publishing, Norwood, New Jersey.