

Perspectives on Usability

Jonas Löwgren

E-mail: jlo@ida.liu.se

Abstract

One of the central concepts in human-computer interaction (HCI) is usability. Interestingly, in spite of its brief history as a scientific and applied discipline, HCI has already produced several different views on usability. These views are, in turn, interrelated with how research and systems development are seen. This paper identifies five different perspectives on usability: general theory, usability engineering, subjectivity, flexibility and sociality. Their interrelations and implications for usability-oriented systems development are discussed.

Perspectives on Usability

Jonas Löwgren

Department of Computer and Information Science
Linköping University, 581 83 Linköping, Sweden
Tel +46 13 281482 • Fax +46 13 142231 • Email jlo@ida.liu.se

ABSTRACT

One of the central concepts in human-computer interaction (HCI) is usability. Interestingly, in spite of its brief history as a scientific and applied discipline, HCI has already produced several different views on usability. These views are, in turn, interrelated with how research and systems development are seen. This paper identifies five different perspectives on usability: general theory, usability engineering, subjectivity, flexibility and sociality. Their interrelations and implications for usability-oriented systems development are discussed.

KEYWORDS: usability, human-computer interaction (HCI), applied HCI.

1. INTRODUCTION

Usability is undoubtedly central to the area of human-computer interaction (HCI). It can be argued that most efforts by HCI researchers and practitioners are ultimately aimed at achieving more usable computer systems. This formulation, however, begs the question of what is really meant by “more usable”. In this paper, I aim at achieving two objectives. The first is to give an overview of the different ways in which actors in HCI have understood the concept of usability. The second, which derives from the first, is to illustrate how these understandings interrelate with different conceptions of the practice of achieving more usable systems.

Lewis (1990) presents an analysis of the conceptual development in the HCI field and uses it as a basis to sketch out a few research themes that might find consensus among the different viewpoints. The following presentation is similar in terms of the viewpoints we both identify, but my intention differs significantly from his. In my mind, an understanding of the conceptual foundations for methods, techniques, tools and other kinds of action-oriented knowledge is a necessary prerequisite for the skilled understanding, adaptation and use of such knowledge. To put this more directly, a system developer is not in a good position to skillfully use, say, a method unless he or she understands the concepts that the method builds on and embodies.

The method I will use to achieve the objectives is interpretative rather than analytical. My presentation of different schools of thought (which I shall call *perspectives* — see below) is based on my own gradual understanding of the area. I will turn to the literature to illustrate key features and concepts, but I have no intention to provide an exhaustive survey of all work pertinent to the topic. My hope is to create a conceptual structure that can aid the student of HCI in understanding the development of the field and its relations with other fields, in

acquiring further knowledge, and in acting skillfully based on the knowledge. It must, however, be noted that the following presentation is my subjective interpretation of data that I have subjectively selected.

With this in mind, let us now turn to the concept of “perspective” that I will use as a structuring principle for the presentation.

1.1 THE CONCEPT OF “PERSPECTIVE”

Simply put, a perspective relates to how somebody observes and understands his or her environment. It can be seen as a filter which causes the observer to select some properties of a phenomenon for consideration and ignore others; it can also be described as a frame for interpretation of the selected properties. Finally, it refers to a standpoint that influences the observer’s choice of position with respect to the phenomenon in question (Nygaard and Sørgaard, 1987). This is the set of connotations I intend when I speak of different perspectives on usability. Moreover, my descriptions will inevitably seem oversimplified. This is intentional, in the same sense that Nurminen (1988) uses in speaking of perspectives as ideal types. By abstracting to a level where we do not speak of particular phenomena or averages, we create concepts that can be used for discussion. To put this another way, we can imagine a continuum between the extreme points represented by my descriptions of the perspectives.

It was stated above that HCI is primarily aimed at achieving more usable systems. Usability is thus one of the goals of HCI activities. I will try to show how the different perspectives influence not only the perception and understanding of usability, but also the actions by which usability is achieved: approaches to systems development.

1.2 OUTLINE

The paper is structured in the following way. After an overview of the five perspectives I have identified and their interrelations, I discuss each of the perspectives in more detail with examples from well-known and easily available literature. The paper closes with a summary and a few concluding remarks on the current state of the discipline.

2. OVERVIEW

The proposed structure consists of five different perspectives on usability, as illustrated in Figure 1. The links between them are meant to illustrate conceptual relations, the precise nature of which should be clear from my descriptions below.

The *general theory* perspective can be characterized (or caricaturized?) as the traditional scientific approach to usability. The aim is to accumulate pieces of knowledge about human interaction with computers. Theories are developed to account for the accumulated data and to predict how humans would interact with new systems.

The general theory perspective proved to be of limited value for HCI practitioners. In the *usability engineering* perspective, the emphasis is on practical applicability rather than generalizability. Usability is operationalized in ways that make it amenable to measurements in iterative software engineering processes. A distinguishing trait of the usability engineering perspective is that the usability of a system under construction is conceptually divorced from its utility.

The *subjectivity* perspective is a fairly strong reaction to the tendency of usability engineering to view usability as a property of the system under construction. Usability, it is argued, must be seen as a strictly subjective quality that emerges only when the system is used by its intended users in their own context. Moreover, utility and usability are seen as inseparable.

Usability engineering is heavily focused on the period of time ranging from project incep-

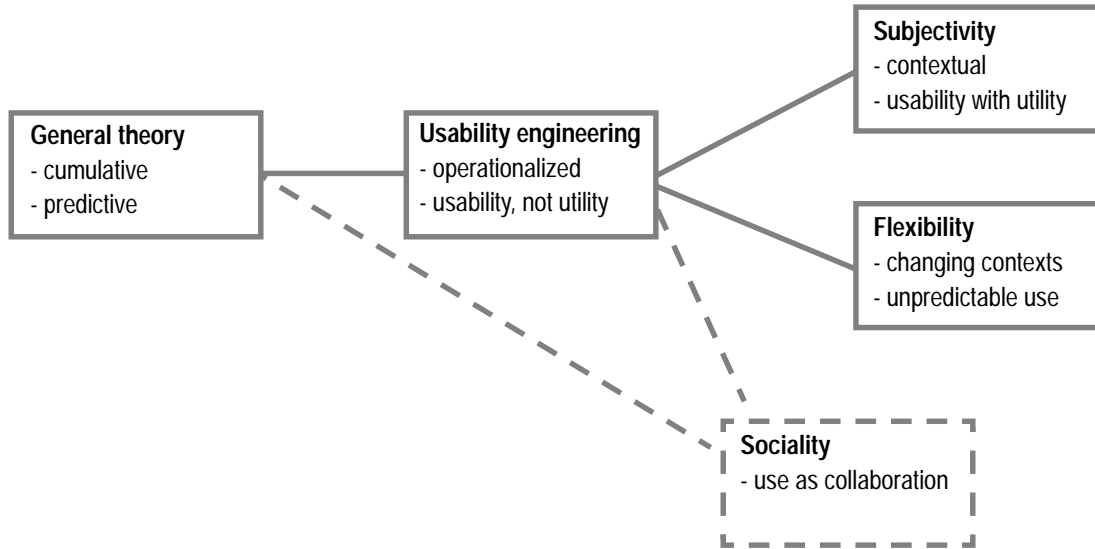


Figure 1: Five perspectives on usability. The progression from left to right roughly approximates a chronological order.

tion to system delivery. The *flexibility* perspective essentially entails a critique of this narrow focus, arguing instead that usability must take into account the dynamic and constantly changing context of ultimate use.

The *sociality* perspective on usability is gradually emerging as computer-supported cooperative work (CSCW) constitutes itself as a scientific field of inquiry into computer use. It is not straightforward to relate it to any of the previous perspectives, but it can probably most fruitfully be understood as a reaction against the traditional view of HCI (particularly in general theory and usability engineering) as one user using one system, more or less in a social vacuum.

3. GENERAL THEORY

The perspective of general theory has its roots in experimental psychology. The phenomenon under study is a human interacting with a computer; the approach is basically positivistic. By performing controlled experiments, we acquire pieces of knowledge about causal relations between independent and dependent variables, e.g., the effects of menu layout on task completion time. By observing and adhering to methodological standards, we know that the knowledge is true within statistical limits. Moreover, it is universally applicable to all instances of human-computer interaction given that the sample population is representative of all users. The amount of studies performed in this way and reported in the literature is considerable; for example, the well-known collection of theory-based design guidelines by Smith and Mosier (1986) contains no less than 173 references, even though it was compiled, fairly selectively, close to 10 years ago.

Creating general theory is more than merely accumulating pieces of knowledge. In order to provide general and useful knowledge, theories are induced from the available data. An induced theory should obviously account for the data, but it also has to be general enough to go beyond it. In other words, a theory can predict outcomes for situations that have never been observed.

3.1 THE PSYCHOLOGY OF HUMAN-COMPUTER INTERACTION

As a highly influential example of general theory, let us consider the 1983 book *The psychology of human-computer interaction* by Card, Moran and Newell. In it, the authors present a cognitive theory of expert-level human interaction with highly interactive systems and specifically text editors. The theory is an extension of information processing psychology, in which the Model Human Processor (MHP) is seen as consisting of three processors (perceptual, cognitive and motor), a working memory and a long-term memory. The performance of the components is characterized in terms of response times and storage capacities. The MHP is further guided by a set of principles of operation. The content being processed, i.e., the user's cognitive structure, consists of four components: goals; operators; methods for achieving the goals; and selection rules for choosing among the methods (hence the famous acronym GOMS).

Card et al. are concerned with the practical applicability of their theory. The intention is that system designers themselves should be able to make use of it in their design work. To this end, "it is necessary that a psychology of interface design be cast in terms homogeneous with those commonly used in other parts of computer science and that it be packaged in handbooks that make its application easy" (p. 13). The latter part of the book describes examples of such packaging in the form of predictive analysis methods on the levels of keystrokes and what is called unit tasks. If it is possible to find suitable packaging, there is no doubt in the authors' minds that theory-based design would be valuable:

The domain of concern to us, and the subject of this book, is how humans interact with computers. A scientific psychology should help us in arranging this interface so it is easy, efficient, error-free—even enjoyable. (Ibid., p. 1)

Looking back, we can summarize the general theory perspective on usability by saying that we can design more usable computers if we accumulate general knowledge of human behavior in HCI. Such knowledge is produced in the form of formal, abstracted and cumulative theory that serves as a tool for the designer's thought (Newell and Card, 1986); it can also be expressed in the form of practical methods for prediction and evaluation.

3.2 APPLICABILITY

The applicability of general theory has, of course, been a continuous source of concern for HCI researchers. The most coherent attempt to address the relations between theory and design is perhaps the notion of *cognitive engineering*. Some authors (e.g., Lansdale and Ormerod, 1994, p. 19) characterize it quite straightforwardly as the application of cognitive psychology to interface design. Woods and Roth (1988) see it as an applied cognitive science that draws on knowledge and techniques from cognitive psychology and related disciplines to provide the basis for principle-driven design. Norman, who allegedly invented the term cognitive engineering (Norman, 1987), questions the unidirectional view of design principles as encapsulated theoretical knowledge. He argues that cognitive engineering — the applied field — should even drive the development of theory in cognitive science. The analysis of direct manipulation in terms of semantic and articulatory directness is given as an example of a "theory of action".

Green (1994) views the relation between (psychological) theory and design as one in which theory has much to offer in terms of understanding and addressing design problems, but is not being used to its potential. To address the problem, he advocates the development of a "boundary vocabulary" of theoretically well-founded concepts and constructs that are mutually acceptable.

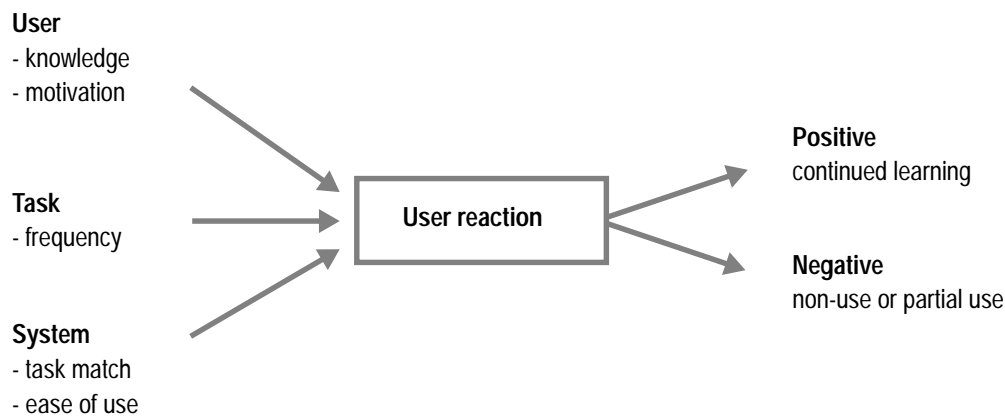


Figure 2: A causal framework for usability (adapted from Eason, 1984).

3.3 THE IMPORTANCE OF USE CONTEXTS

The general theory perspective is influenced by traditional experimental paradigms, which typically entail fairly short experiments in laboratory settings. It can be argued that many crucial usability issues surface only in real-world situations, where system use occurs over extended periods of time and interacts with all the user's other work tasks. In a 1984 paper, Ken Eason concludes from a number of field studies of systems in use that an alternative to traditional experiments is necessary (Eason, 1984). He points out in the introduction that the concept of usability is not susceptible to scientific investigation, as it lacks definition and has no agreed operational form of measurement. Moreover, many of the variables that influence usability are often excluded from experimental paradigms with the result that usability issues are not evident in the result of experimental studies. To address these shortcomings, Eason proposes a causal framework to summarize the variables affecting the usability of a system in use (Figure 2).

The independent variables in the framework are of three kinds: user, task and system. *User* variables describe who actually uses the system, her levels of computer and task knowledge, her degree of discretion in using the computer, her level of motivation, and so on. *Tasks* that the user carries out are characterized by frequency, openness, etc. Finally, the *system* is described as the primary independent variable which we can manipulate to improve usability. It is defined in terms of how well its functions match the user's tasks and how easy it is to learn and use.

The dependent variable of the framework is the user's reaction, which Eason describes as being formed by means of an implicit cost-benefit analysis. Over time, the user accumulates a knowledge base of task-system connections as the system is used in a sequence of task episodes. The emerging strategy for use may represent a positive outcome in which the user locates and uses appropriate system functions for every new task and progressively masters the system. The reverse scenario occurs when negative outcomes prevail and use of the system is discontinued. Eason points out, based on his field studies, that under realistic conditions the user appears to approach a state of equilibrium where further learning about the system is minimized.

For our purposes, the main point of the causal framework is its implications for the study of

usability. Eason envisions a field-oriented methodology that preserves the realistic, total user-task-system context. A hypothesis in this framework is a contingency statement that relates a user, task and system with certain characteristics to observed extent of use, and learning (or non-learning) strategy. Such hypotheses are investigated in field studies or simulations. As more data points are collected, it is possible to use multivariate analysis to abstract out certain contingencies with predictive power beyond the original data. Such abstractions, i.e., theories, can be used to formulate further hypotheses for investigation. The analysis may also provide grounds for refining the framework itself.

Eason's framework can be seen as an early attempt to address important questions concerning laboratory settings versus real use contexts within the perspective of general theory. However, not many researchers accepted the invitation to further refine the framework or fill it with empirical contents.

4. USABILITY ENGINEERING

In spite of the ambitious aims, general theory of HCI did not have the kind of impact on professional software development contexts that its proponents had perhaps hoped for. Whether the packaging was wrong or the intended audience was not receptive enough is irrelevant for our purposes. The fact remains that a new perspective emerged from the "failure", drawing on the methodological foundations of general theory but orienting itself towards the classical paradigm of applied science: engineering.

Usability engineering is a process, grounded in classical engineering, which amounts to specifying, quantitatively and in advance, what characteristics and in what amounts the final product to be engineered is to have. This process is followed by actually building the product, and demonstrating that it does indeed have the planned-for characteristics.

Engineering is not the process of building a perfect system with infinite resources. Rather, engineering is the process of economically building a working system that fulfills a need. Without measurable usability specifications, there is no way to determine the usability needs of a product, or to measure whether or not the finished product fulfills those needs. If we cannot measure usability, we cannot have a usability engineering.

(Good et al., 1986, p. 241)

Usability engineering as a systems development process is typically described as consisting of three main steps. The first is a user and task analysis, in which the intended user population is investigated along with the work tasks the system is intended to support. Next, a usability specification is negotiated in analogy with conventional requirement specifications. The usability specification details a number of measurable usability goals (see below). This specification is used as the control instrument for an iterative process, in which a prototype is designed and usability-tested. If it fulfills the specified goals, the design is adequate with respect to usability. If not, information for redesign has to be elicited, e.g., from the users participating in the usability tests.

4.1 USABILITY OPERATIONALIZED

This generic usability engineering process addresses a very real problem in software development, namely how to know when to stop iterating. Before usability engineering was articulated, professional software developers had already known for a long time that user requirements are not like computational space or complexity constraints. To specify the requirements first and then develop a system fulfilling them would be feasible for the latter category, but not for the former. Common practice dictated iterative development, where a

prototype was designed, negotiated with the users and redesigned (Hammond et al., 1983). There was, however, always the risk of iterating without being able to demonstrate progress in the direction of fulfilling the user requirements. The solution offered by usability engineering was to operationalize usability, i.e., to define it in terms of measurable quantities. Examples are

- user performance on specified tasks, measured in terms of task completion rate, completion time or number of errors;
- flexibility of the design, measured in terms of the proportion of users in a heterogeneous population able to perform the test tasks;
- learnability, measured in completion rate, time, errors, or use of documentation or help desk;
- user's subjective preference or degree of satisfaction.

These and many other examples of operationalized usability definitions can be found in the literature (e.g., Carroll and Rosson, 1985; Shackel, 1986; Whiteside et al., 1988; Dumas and Redish, 1993; Löwgren, 1993; Nielsen, 1993).

To a great extent, usability engineering is about generalizability. However, generalizations are not *absolute* (within statistical limits) as in the general theory perspective, but rather *adequate*. Results from user and task analysis are assumed to give an adequate picture of the intended use context. Usability tests are performed to assure adequate usability once the system moves into the real use context.

The adequate generalizability of usability engineering makes usability an objective (or rather, an adequately objective) property of the system under construction. For instance, usability testing a prototype may tell us that on the average, the system allows an adequately representative user to learn an adequately representative task in 5 minutes while committing 2 errors. If the usability specification states that 1 error is the goal to be reached, the only degree of freedom we have is the design of the system. The redesigned prototype may score 5 minutes and 0.5 errors, allowing us to conclude that the usability goal has been satisfied. Thus, the "usability value" depends only on the design of the system. The further conceptual step of regarding usability as a property of the system is neither far-fetched nor uncomfortable.

4.2 USABILITY AND UTILITY

Seeing usability as an adequately objective property of the system also entails divorcing it from the utility of the system (Nielsen, 1993, is particularly explicit on this topic). It is as if usability engineering is concerned with the users' adequately efficient and error-free access to the services of the system, but not at all with the appropriateness of the services themselves. This divorce can be explained by the simple observation that it is impossible to formulate adequately objective utility goals (beyond the obvious quantitative self-estimates). Grudin (1993), however, offers an alternative explanation. He points out that usability engineering emerged in professional product development contexts, where the services of the product are decided by managers and marketing, long before the developers are given their brief. Hence, the utility of the system — which is determined by the appropriateness of the services — is not for the developers to concern themselves with.

To summarize, the usability engineering perspective sees usability as a measurable and adequately objective property of the system under development. Moreover, usability can be addressed and achieved without considering the utility of the system.

5. SUBJECTIVITY

The notion of usability as an adequately objective property can, of course, be questioned on the grounds of whether even adequately general knowledge is possible to acquire or sensible to expect. Moreover, usability engineering critics voiced concerns that a strong focus on measurability always runs the risk of concentrating on easily measurable aspects at the expense of perhaps more important ones. Two of the most influential actors within the usability engineering perspective, John Whiteside and Dennis Wixon, were also among the first to clearly articulate its limitations. In a short but intense 1987 paper, they wrestle with the shortcomings of regarding usability as an adequately objective property.

For us, software usability refers to the extent to which software supports and enriches the ongoing experience of people who use that software. This direct emphasis on experience is at variance with 'standard' definitions of usability that concentrate on, as primary, measurable manifestations of usability such as productivity, preference, learnability, or throughput. Although these may be interpreted as important properties of usability, for us they are not primary. Primary is the person's experience at the moment experienced.

(Whiteside and Wixon, 1987, p. 18)

Thus, usability becomes a purely subjective property of the interaction between a specific user and the computer at a specific moment in time. Moreover, there is no question that usability in the subjectivistic perspective also comprises utility.

If the [usability] goals are not grounded in something really meaningful to the users, the resulting product will be useless to them. (Ibid., p. 20)

It is clear that the subjectivistic perspective rules out any attempts to specify usability goals in a way that purports to generality across the user population. How, then, can we develop usable systems? The answer from Whiteside, Wixon and colleagues emerged gradually over several years in the form of contextual design (Whiteside et al., 1988; Wixon et al., 1990; Holzblatt and Beyer, 1993; Holzblatt and Jones, 1993; Beyer and Holzblatt, 1995).

5.1 CONTEXTUAL DESIGN

Essentially, contextual design is described as a cyclic process of requirements generation, design, implementation and evaluation, similar to Boehm's (1986) spiral model. A suite of techniques are used to ensure that the developers understand the users and customers, and that this understanding is reflected in the system. A contextual data collection method called *contextual inquiry*, based on ethnographic field research techniques and a notion of developer-as-apprentice, is used to acquire the necessary understanding. Contextual inquiry concentrates on observations of users at work and discussions with them in the context of the ongoing tasks. *Affinity diagrams* and graphical *work modelling languages* are used to produce a coherent interpretation of the collected data. The work is *redesigned* in a participatory effort and evaluated using further contextual inquiry. The *conceptual structure* of the system is designed, followed by a mapping to the *user interface*. The conceptual design is tested using mock-ups; as the shared understanding grows, the prototypes can be more detailed. The emerging understanding also prompts further contextual inquiry and work redesign in an iterative manner.

As should be apparent, there are striking similarities between the development approach growing out of a subjectivist perspective on usability, and what the information systems community has termed participatory design (see, e.g., Bjercknes et al., 1987; Greenbaum and Kyng, 1991; Schuler and Namioka, 1993; a historical overview is given by Clement and Van den

Besselaar, 1993) or the “Scandinavian school” (Floyd et al., 1989). The respect for the uniqueness of each use situation, the concern for work design rather than mere computer system design, and the use of user-accessible design techniques and expressions all closely relate contextual design to participatory design.

This superficial similarity is particularly interesting considering the radically different origins of the two approaches. Participatory design grew out of strong trade union movements and the desirability of democratic participation and skill enhancement in the workplace in Scandinavia in the 70s (Ehn, 1992). Essentially, participatory design has political roots and connotations, whereas contextual design emerged from the inadequacy of an engineering approach to usability. However, in analogy with Grudin’s (1992) comment, they both appear to have the potential for parallel consideration of utility and usability.

6. FLEXIBILITY

In the last years, there has been a growing concern with the inertia of usability engineering and similar processes where it is assumed that usability can be achieved through careful analysis followed by iterative design towards a fixed specification. One part of the argument is that such processes are implicitly based on a mechanistic and static view of the users’ work, whereas contemporary working situations change and develop rapidly and unpredictably. Thus we cannot capture at specification-writing time all the aspects of future situations in which the users will need to use the system.

The view of the users and their work can also be debated from another angle. If we regard the users as cognitively and socially active, continuously growing and developing in their work, then usability by necessity becomes an evolving or emerging property of long-term system use. Laboratory experiments and usability tests are clearly not capable of addressing this conception of usability. The preoccupation of systems development approaches with delivery deadlines will also be questioned; usability work must concentrate on providing process as well as product support for long-term user growth and development.

In a programmatic paper, Adler and Winograd (1992) take the consequences of this view and formulate what I have termed the flexibility perspective on usability.

To break free of the prevalent myths and go beyond current practice, we must articulate new criteria of usability that are appropriate to the tasks of modern computer-based system design and the interwoven tasks of work design... The key criterion of a system’s usability is the extent to which it supports the potential for people who work with it to understand it, to learn and to make changes. Design for usability must include design for coping with novelty, design for improvisation and design for adaptation. (Ibid., p. 7)

6.1 PROCESS FLEXIBILITY

The question is how we develop systems that exhibit usability-as-flexibility. Adler and Winograd concentrate on a view of usability assurance as a dialogue between many parties, pointing out the obvious need for usability efforts early in the development process. They also advocate the use of multiple representations, such as mock-ups, profession-oriented languages and scenarios, to create an equally accessible “extended language” for the communication with potential users. It is clear that they are thinking about the development process in terms that are very similar to those of participatory design. Among the fundamental tenets of participatory design is the view of the development process as one of constant change, in which the developers act as catalysts in initiating the change process and helping to shape its direction. There is no firm deadline by which the system is finished. It seems clear that this

way of looking at systems development enables us to come to terms with design for new and unexpected situations. A recent example is the case study by Kjær and Halskov Madsen (1995), where the authors illustrate how flexibility requirements were addressed in a development process by using participatory design techniques.

6.2 PRODUCT FLEXIBILITY

If we shift focus from process to product, it is interesting to note the increasing interest in systems that the users tailor, extend and modify to suit the emerging needs. The paradigmatic example of this area, which has been termed *tailorability*, *end-user programming* or *end-user computing*, is of course the spreadsheet which possesses several characteristics seen as crucial for usability-as-flexibility: the data representation is concrete and visible, feedback is immediate, and — most importantly — it only takes a few hours of work for the user to construct simple functioning programs that model the problems of interest. Thus, the main effect is motivational rather than cognitive in the sense that the users are not intrinsically interested in programming, but rather in their work domain (Nardi and Miller, 1990).

The design concept of flexibility-in-use is by no means limited to financial calculations; many recent software products, primarily in the office, feature some sort of flexibility. Macro recording and replay, scripting, plug-in modules, tailorable menus and individual preferences are all examples of techniques aimed at providing the user with the flexibility to cope with changing situations. Early examples include the scripting capabilities of the Buttons system in an office automation context (MacLean et al., 1990) and the tailorable domain-oriented design environments advocated by Fischer and Girgensohn (1990). More systematically, Henderson and Kyng (1991) identify three types of tailoring or design-in-use:

- choosing between anticipated behaviors, as in setting the default font of a word processor;
- constructing new behaviors from existing pieces, e.g., recording and using macros;
- modifying the artifact itself.

Henderson and Kyng move to outlining a development cycle of design-in-use in order to identify activities that need to be considered: the tailor creates changes, validates them, packages them as objects, and makes them available; the prospective user inspects, then learns the new objects and finally provides feedback. However, problems may also be anticipated, including the difficulty of providing technical support, the learning required, the risk that an overwhelming array of features will be created and the possibly missing design competence of the tailor.

It seems reasonable to assume that usability-as-flexibility or continued design in use will be increasingly addressed also in bespoke systems development. This would, however, require reconsidering the common notion of “getting it right the first time”.

7. SOCIALITY — AN EMERGING PERSPECTIVE ON USABILITY?

Human-computer interaction, and particularly work within the general theory perspective, has traditionally focused its inquiry on the dyad of one user using one computer system, more or less pulled out of the social context in which this use typically takes place. The emerging research area known as computer-supported cooperative work (CSCW) has instead taken as its point of departure the nature of cooperation, the social organization of work and how computers can be used to support it. In a programmatic statement, the main concern of CSCW is said to be

... an endeavor to understand the nature and requirements of cooperative work with the objective of designing computer-based technologies for cooperative work arrangements.

(Schmidt and Bannon, 1992, p. 11)

For our purposes — to explore the concept of usability — Bannon's (1991) call for a widening of the focus to encompass groups of people and machines engaged in collaborative tasks can certainly be relevant. Indeed, the recent CSCW literature contains studies of cooperative, computer-supported work that can easily be interpreted in terms of what we may call social or organizational usability. For example, Abbott and Sarin (1994) report a case study of a workflow management system in use where "usability issues and challenges" are seen to include, e.g., support for interworker coordination and organizational policy management.

If usability is seen as social, what are the implications for the practice of achieving more usable systems? One striking observation is the interest in recent years in reconciling sociological inquiry methods, and particularly ethnography, with systems development. The argument is briefly that we need ways of understanding work as a social phenomenon if we want to support it with computer technology. Ethnography is aimed at understanding and analyzing the sociality of settings under study (such as work settings). Hence it should have a place in the development of "socially usable" systems. It is yet too early to say whether there is substance to this claim. Arguments in favor include, e.g., discussions of the pragmatic ways in which ethnography can play significant roles in CSCW system design (Hughes et al., 1994): recommendations include *concurrent ethnography* to inform design through different stages; *quick-and-dirty ethnography* to gain a first overview of the work domain before design commences; *evaluative ethnography* to validate design proposals; the *re-examination of previous studies* to inform early design. On the other hand, critics have cautioned against the view of ethnography as a data collection technique, instead pointing to the value of "analytic ethnography" (Anderson, 1994), an "ethnographic orientation" (Cooper et al., 1995) or "foundational analysis" (Nyce and Löwgren, 1995) where normally taken-for-granted concepts are challenged and radically novel design possibilities are brought to light.

8. CONCLUDING REMARKS

In this section, I summarize some salient features of the five perspectives and then briefly discuss the state of the HCI discipline with respect to the perspectives. The paper is then concluded by revisiting the originally stated objectives.

8.1 SUMMARY

Bearing in mind that the perspectives I have introduced are ideal types rather than representative prototypes, let us summarize what implications they have for the concept of usability. In table 1, three questions are asked of each of the perspectives: Where does usability "reside", i.e., what is usability a property of? Where do we study usability? How do we build usable systems, i.e., how do we achieve usability?

	Usability is a property of...	Usability is studied in...	Usability is achieved by ...
General Theory	user-system interaction	experimental laboratories	knowledge of user behavior
Usability engineering	user interface of system	usability laboratories	specification and iteration
Subjectivity	individual use situations	real use contexts	design for unique use situations
Flexibility	long-term use	real use contexts	continued design in use
Sociality	social use situations	labs or real use contexts	

Table 1: A summary of the five perspectives.

The table should be fairly easily traceable to my characterizations above, except for the vague characterization of the sociality perspective. The reason is that the field of CSCW is extremely diverse and active. Unlike, e.g., the subjectivity perspective which I have constructed as a reaction to usability engineering, the sociality perspective has different roots and usability *per se* is not seen as an important concept. Lab studies occur in the literature next to studies of real use contexts, objective next to subjective epistemologies. My reason for including the sociality perspective at all is that I find it helpful in understanding an important direction in the development of “more usable systems”: from human factors to human actors, to borrow a phrase from Bannon (1991). It may also illuminate the relations between HCI and CSCW.

Two more general observations can be made concerning the conceptual structure I have introduced. The first is that the relations between the different perspectives are fairly intricate. A few distinctive traits are given below.

- **General theory — Usability engineering.** The basic assumptions are preserved: the laboratory is maintained, and “representative users” still exist. The distinctive feature of the relation is the rewriting of the experimental paradigm and the concept of usability to suit the business needs of professional software development.
- **Usability engineering — Subjectivity.** The shift from adequately objective knowledge of usability to purely subjective has far-ranging implications for how usability-oriented development is viewed. The notion of freezing a specification early on is discarded, along with the usability lab as an approximation of real use. The users become work experts rather than computer novices.
- **Usability engineering — Flexibility.** Here, the big leap is to question the view of systems development as a process that starts with a brief and ends when the system is delivered. The key concept of the flexibility perspective, be it in terms of process or product, is that development continues for as long as the system is used. The implications for systems development methodology are considerable.
- **Sociality — The rest.** As stated above, the sociality perspective may be hard to relate directly to the others. However, an important concept in relating sociality to general theory and usability engineering is the view of the users as fundamentally social beings. This is at variance with the view of the user as a lonely performer of isolated tasks.

The other general observation is simply that these different perspectives on usability, seen as conceptual development, entail the orientation of HCI in relation to other fields. A good example is to be found in the subjectivity and flexibility perspectives, and particularly in the emerging resonance with participatory design and other developments in the field of information systems. This is no coincidence: the larger context of computer system use is, of course, a necessary condition for the occurrence of human-computer interaction.

8.2 STATE OF THE DISCIPLINE

To restate a starting point for my presentation, the perspectives I have outlined are simplified abstractions of different ways to view usability and usability-oriented systems development. They are not mutually exclusive; rather they coexist, more or less peacefully, in the HCI field. For example, the proceedings of the latest CHI conference from May 1995 contains contributions exemplifying all of the five perspectives: Bauer and John (1995) illustrate general theory; Bradley and Johnk (1995) use a usability engineering approach; Karat and Dayton (1995) illustrate subjectivity; Malinowski and Nakakoji (1995) represent the flexibility view; Harper and Sellen (1995) address the sociality of work.. A very coarse summary of the current state of each of the perspectives would be as follows.

- General theory is being extended and developed in psychologically based research. Theoretical findings are often combined with practical design guidelines to provide the link from theory to design (a good example is Mayhew, 1992). Tools are developed to further the practical use of theory and guidelines, either by providing search and browse capabilities or by automating design or evaluation. Applied psychology, often presented as cognitive engineering (see above), is used as a teaching device.
- Usability engineering and the concept of the usability lab has gained wide acceptance in professional software development practice (see Wiklund, 1994). A reason for this may be that it fits well with contemporary business structures.
- The subjectivity perspective is consolidated as the interest grows within the HCI community in contextually rich approaches to systems development and the intersections with fields such as information systems. Even though it has so far mainly attracted researchers, commercial approaches are beginning to emerge that treat usability as subjective, typically within a larger framework of organizational development.
- Flexibility on a small scale is ubiquitous in off-the-shelf software. It is not unlikely that the concept of product flexibility will move into bespoke systems development as well. To start addressing process flexibility, however, would require reconsidering the project orientation, business structure and other foundations of professional systems development.
- As I have already indicated, the sociality perspective is hard to summarize and assess. However, it may be safe to say that serious consideration of the social context of computer use forces us to question much of what has hitherto been done in HCI.

In conclusion, do we now know what is meant by achieving “more usable” systems? I think the answer has to be that it depends. This presentation has been intended to give a broader understanding of what it depends on, and how the dependencies play themselves out in usability-oriented systems development.

What leverage can be gained from such an understanding? Distanced and academic as it may seem, it is my hope that the conceptual framework I present can facilitate more well-informed acquisition of actionable knowledge — methods, techniques, tools, etc — and reflection upon its appropriate use. In the long term, such reflection is necessary for the competent practice and evolution of usability-oriented systems development.

ACKNOWLEDGEMENTS

I am indebted to Pär Carlshamre, Lena Holmberg, Theis Meggerle and Odd Steen for comments that helped me shape my thoughts and the way to present them.

REFERENCES

- Abbott, K., and Sarin, S. (1994). Experiences with workflow management: Issues for the next generation. In *Proc. Conf. Computer Supported Cooperative Work (CSCW'94)*, pp. 113-120. New York: ACM Press.
- Adler, P., and Winograd, T. (1992). The usability challenge. In Adler, P., and Winograd, T. (eds) *Usability: Turning technologies into tools*, pp. 3-14. New York: Oxford University Press.
- Anderson, R. (1994). Representations and requirements: The value of ethnography in system design. *Human-Computer Interaction* 9(2):151-182.
- Bannon, L. (1991). From human factors to human actors: The role of psychology and human-computer interaction studies in system design. In Greenbaum and Kyng (op cit), pp. 25-44.
- Bauer, M., and John, B. (1995). Modeling time-constrained learning in a highly interactive task. In *Human Factors in Computing Systems (CHI'95 Proceedings)*, pp. 19-26. New York: ACM Press.
- Beyer, H., and Holzblatt, K. (1995). Apprenticing with the customer. *Communications of the ACM* 38(5):45-52.
- Bjerknes, G., Ehn, P., and Kyng, M. (1987). *Computers and democracy: A Scandinavian challenge*. Aldershot: Avebury.
- Boehm, B. (1986). A spiral model of software development and enhancement. *IEEE Computer* 21(5):61-72.
- Bradley, R., and Johnk, L. (1995). Replacing a networking interface "from hell". In *Human Factors in Computing Systems (CHI'95 Proceedings)*, pp. 538-545. New York: ACM Press.
- Card, S., Moran, T., and Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum.
- Carroll, J., and Rosson, M. (1985). Usability specifications as a tool in iterative development. In Hartson, H. (ed) *Advances in human-computer interaction*, pp. 1-28. Norwood, NJ: Ablex.
- Clement, A., and Van den Besselaar, P. (1993). A retrospective look at PD projects. *Communications of the ACM* 36(4):29-37.
- Cooper, G., Hine, C., Rachel, J., and Woolgar, S. (1995). Ethnography and human-computer interaction. In Thomas, P. (ed) *The social and interactional dimensions of human-computer interfaces*, pp. 11-36. Cambridge: Cambridge University Press.
- Dumas, J., and Redish, J. (1993). *A practical guide to usability testing*. Norwood, NJ: Ablex.
- Eason, K. (1984). Towards the experimental study of usability. *Behaviour and Information Technology* 3(2):133-143.
- Ehn, P. (1992). Scandinavian design: On participation and skill. In Adler, P., and Winograd, T. (eds) *Usability: Turning technologies into tools*, pp. 96-132. New York: Oxford University Press.
- Fischer, G., and Girgensohn, A. (1990). End-user modifiability in design environments. In *Human factors in computing systems (CHI'90 Proceedings)*, pp. 183-191. New York: ACM Press.
- Floyd, C., Mehl, W.-M., Reisin, F.-M., Schmidt, G., and Wolf, G. (1989). Out of Scandinavia: Alternative approaches to software design and system development. *Human-Computer Interaction* 4:253-350.
- Good, M., Spine, T., Whiteside, J., and George, P. (1986). User-derived impact analysis as a tool for usability engineering. In *Human factors in computing systems (CHI'86 Proceedings)*,

- pp. 241-246. New York: ACM Press.
- Green, T. (1994). Why software engineers don't listen to what psychologists don't tell them anyway. In Gilmore, D. et al. (eds) *User-centred requirements for software engineering environments*, pp. 323-333. Berlin: Springer-Verlag.
- Greenbaum, J., and Kyng, M. (1991). *Design at work: Cooperative design of computer systems*. Hillsdale, NJ: Lawrence Erlbaum.
- Grudin, J. (1992). Utility and usability: Research issues and development contexts. *Interacting with Computers* 4(2):209-217.
- Grudin, J. (1993). Obstacles to participatory design in large product development organizations. In Schuler and Namioka (op. cit.), pp. 99-119.
- Harper, R., and Sellen, A. (1995). Collaborative tools and the practicalities of professional work at the International Monetary Fund. In *Human Factors in Computing Systems (CHI'95 Proceedings)*, pp. 122-129. New York: ACM Press.
- Hammond, N., Jørgensen, A., MacLean, A., Barnard, P., and Long, J. (1983). Design practice and interface usability: Evidence from interviews with designers. In *Human factors in computing systems (CHI'83 Proceedings)*, pp. 40-44. New York: ACM Press.
- Henderson, A., and Kyng, M. (1991). There's no place like home: Continuing design in use. In Greenbaum and Kyng (op cit), pp. 219-240.
- Holzblatt, K., and Beyer, H. (1993). Making customer-centered design work for teams. *Communications of the ACM* 36(10):93-103.
- Holzblatt, K., and Jones, S. (1993). Contextual inquiry: A participatory technique for system design. In Schuler and Namioka (op. cit.), pp. 177-210.
- Hughes, J., King, V., Rodden, T., and Andersen, H. (1994). Moving out from the control room: Ethnography in system design. In *Proc. Conf. Computer Supported Cooperative Work (CSCW'94)*, pp. 429-439. New York: ACM Press.
- Karat, J., and Dayton, T. (1995). Practical education for improving software usability. In *Human Factors in Computing Systems (CHI'95 Proceedings)*, pp. 162-169. New York: ACM Press.
- Kjær, A., and Halskov Madsen, K. (1995). Participatory analysis of flexibility. *Communications of the ACM* 38(5):53-60.
- Lansdale, M., and Ormerod, T. (1994). *Understanding interfaces: A handbook of human-computer dialogue*. London: Academic Press.
- Lewis, C. (1990). A research agenda for the nineties in human-computer interaction. *Human-Computer Interaction* 5(2-3):125-143.
- Löwgren, J. (1993). *Human-computer interaction: What every system developer should know*. Lund: Studentlitteratur.
- MacLean, A., Carter, K., Lövstrand, L., and Moran, T. (1990). User-tailorable systems: Pressing the issues with Buttons. In *Human factors in computing systems (CHI'90 Proceedings)*, pp. 175-182. New York: ACM Press.
- Malinowski, U., and Nakakoji, K. (1995). Using computational critics to facilitate long-term collaboration in user interface design. In *Human Factors in Computing Systems (CHI'95 Proceedings)*, pp. 385-392. New York: ACM Press.
- Mayhew, D. (1992). *Principles and guidelines in software user interface design*. Englewood Cliffs, NJ: Prentice Hall.

- Nardi, B., and Miller, J. (1990). The spreadsheet interface: A basis for end user programming. In Diaper, D. et al. (eds) *Human-computer interaction — Interact '90*, pp. 977-983. Amsterdam: North-Holland.
- Newell, A., and Card, S. (1986). Straightening out softening up: Response to Carroll and Campbell. *Human-Computer Interaction* 2(3):251-267.
- Nielsen, J. (1993). *Usability engineering*. Boston: Academic Press.
- Norman, D. (1987). Cognitive engineering — Cognitive science. In Carroll, J. (ed) *Interfacing thought: Cognitive aspects of human-computer interaction*, pp. 325-336. Cambridge, MA: MIT Press.
- Nurminen, M. (1988). *People or computers: Three ways of looking at information systems*. Lund: Studentlitteratur.
- Nyce, J., and Löwgren, J. (1995). Towards foundational analysis in human-computer interaction. In Thomas, P. (ed) *The social and interactional dimensions of human-computer interfaces*, pp. 37-47. Cambridge: Cambridge University Press.
- Nygaard, K., and Sørgaard, P. (1987). The perspective concept in informatics. In Bjerknes et al. (op cit), pp. 371-393.
- Schmidt, K., and Bannon, L. (1992). Taking CSCW seriously: Supporting articulation work. *Computer Supported Cooperative Work* 1(1-2):7-40.
- Schuler, D., and Namioka, A. (1993). *Participatory design: Principles and practices*. Hillsdale, NJ: Lawrence Erlbaum.
- Shackel, B. (1986). Ergonomics in design for usability. In Harrison, M., and Monk, A. (eds) *People and computers: Proc. 2nd conf. BCS HCI specialist group*, pp. 45-64. Cambridge: Cambridge University Press.
- Smith, S., and Mosier, J. (1986). Guidelines for designing user interface software. Report ESD-TR-86-278, Mitre Corp., Bedford, Mass.
- Whiteside, J., and Wixon, D. (1987). The dialectic of usability engineering. In Bullinger, H.-J., and Shackel, B. (eds) *Human-computer interaction — Interact '87*, pp. 17-20. Amsterdam: North-Holland.
- Whiteside, J., Bennett, J., and Holzblatt, K. (1988). Usability engineering: Our experience and evolution. In Helander, M. (ed) *Handbook of human-computer interaction*, pp. 791-817. Amsterdam: Elsevier.
- Wiklund, M. (1994). *Usability in practice: How companies develop user-friendly products*. Boston: AP Professional.
- Wixon, D., Holzblatt, K., and Knox, S. (1990). Contextual design: An emergent view of system design. In *Human factors in computing systems (CHI'90 Proceedings)*, pp. 329-336. New York: ACM Press.
- Woods, D., and Roth, E. (1988). Cognitive engineering: Human problem solving with tools. *Human Factors* 30(4):415-430.