SOFTWARE PROCESS IMPROVEMENT AND PRACTICE
Softw. Process Improve. Pract. 2003; **8**: 111–131 (DOI: 10.1002/spip.174)

The Usability Design Process – Integrating User-centered Systems Design in the Software Development Process

Research Section

Bengt Göransson[1,2*,†], Jan Gulliksen[2] and Inger Boivie[2]
[1] *Enea Redina AB, Smedsgränd 9, SE-75320 Uppsala, Sweden*
[2] *Department of Information Technology, Division of HCI, Uppsala University, PO Box 337, SE-75105 Uppsala, Sweden*

This article reviews current efforts in bridging the gaps between software engineering and Human–Computer Interaction (HCI) and describes some critical issues that must be resolved in order to reconcile some of the differences between the two fields. We argue that user-centered systems design (UCSD) must be tightly integrated in the software development process and suggest the usability design process as a way of doing this. The usability design process is a UCSD approach for developing usable interactive systems, combining usability engineering with interaction design, and emphasizing extensive active user involvement throughout the iterative process. We outline the usability design process and illustrate the steps in the process with examples from real-life design cases. Finally, we provide an example of how the usability design process can be implemented in a commercial software-development process, Rational Unified Process™ (RUP). Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: user-centered systems design; usability design; usability; interaction design; software development process; Rational Unified Process

1. INTRODUCTION

Usability has long been a major concern within software development (see for instance, Wasserman 1981, Gould and Lewis, 1983, Wasserman *et al*. 1986). Even so, there is still a large number of software products or systems with poor usability. Human–Computer Interaction (HCI) emerged as a field of research with the purpose of studying the interaction between the user and computer technology. Over the years, the HCI field and software engineering (SE) have developed substantial and separate bodies of knowledge, making it difficult for the individual software engineer to acquire extensive knowledge within HCI and for the usability professional to become a skilled software engineer. To software engineers, usability is just one of the many aspects that must be taken into account in the development process – if time permits. Moreover, they tend to consider usability the responsibility of somebody else (Boivie *et al*. 2003). Many usability professionals, on the other hand, are skilled at performing user studies and evaluations, but have little experience of programming

* Correspondence to: Bengt Göransson, Enea Redina AB, Smedsgränd 9, SE-75320 Uppsala, Sweden
† E-mail: Bengt.Goransson@enea.se

and software engineering. This is unfortunate. Usability professionals should, on the contrary, be deeply involved in the software-development process, contributing directly to the artifacts produced, or they will have little impact on the resulting systems/products. We therefore believe that it is important to bring closer together the fields of software engineering and HCI. There are several efforts to do so, for instance, the development of tools and techniques within software engineering, e.g. use cases (Fowler 1997, Jacobson *et al*. 1999), design patterns (Gamma *et al*. 1995) or new types of object models (Hudson 2001, Roberts *et al*. 1998). Similar efforts have been made within the HCI field, for instance, scenario-based techniques (Carroll 2000), user-centered requirements engineering (Sutcliffe 2002) and usability engineering (Nielsen 1993).

Many of these techniques and tools are doubtlessly effective within their contexts, but we are concerned with the development process from a life-cycle perspective (ISO/TR 18529 2000). To integrate usability into the software development process, and for usability professionals to be deeply involved in software development, we need a *process* for usability. Software engineering has produced various different software-development processes, with more or less focus on usability, for instance, Rapid Application Development (RAD). Currently, in Sweden, the Rational Unified Process™ (RUP) (Kruchten 1998) is the most widely used commercial software-development process. The RUP is explicitly architecture-centered and provides little support for usability matters. The use case approach used in the RUP has been criticized for its lack of usability focus and poor support for user interface design (Constantine and Lockwood 1999, Gulliksen *et al*. 2001). Working with usability is quite difficult under such circumstances, given that usability work is not explicitly promoted or supported in the development processes. What we need is a usability process that can be seamlessly integrated into the software-development process, shifting the focus of the development process towards usability and extensive user involvement.

2. PROCESSES FOR USABILITY – ENGINEERING VERSUS DESIGN

Usability as a concept has been fairly well defined in ISO 9241-11 (1998). This definition prescribes an engineering approach, where usability is specified and measured in terms of usability metrics, i.e. quantitative and measurable goals for effectiveness, efficiency and satisfaction (Whiteside *et al*. 1988). We believe that usability goals are essential for maintaining a focus on usability throughout the process, but in our practical experience – from a large number of projects in different organizations – usability goals are difficult to specify. Usability metrics contribute to the engineering nature of usability work and software development. According to the dictionary, engineering means the 'application of scientific and mathematical principles to practical ends such as the design, manufacture, and operation of efficient and economical structures, machines, processes, and systems' (www.dictionary.com). Within software engineering, the 'engineering aspect' is of great importance – the software-development process is viewed as a construction process, similar to those used in, for instance, civil engineering. Design, on the other hand, is 'a creative activity that involves bringing into being something new and useful that has not existed previously' (Jones 1981). There is an on-going discussion about the extent to which design processes and practices can be specified, described and thereby communicated and made explicit to others (see for instance Fallman 2003, for a recent account). There is, however, not much evidence that design can be made independent of talent or all aspects involved in craft work (Wallace and Andersson 1993, Lewis 1990). Nor is there much evidence that structured design processes are successful in capturing the way designers actually work, or provide support for such work (Fallman 2003). Efforts have, nevertheless, been made to specify design as an engineering task, for example,

> *'Engineering design is the use of scientific principles, technical information and imagination in the definition of a mechanical structure, machine or system to perform prespecified functions with maximum economy and efficiency. Design refers to both the process of developing a product, artifact or system and to the various representations (simulations or models) of the product that are produced during the design process'* (Preece *et al*. 1994).

Obviously, it is difficult to reconcile the engineering approach, relying on structured processes, and the creative aspects of design. Can a skilled interaction designer replace a well-defined design process, for instance, the usability engineering life-cycle process (Mayhew 1999)? One of the main differences between the two approaches is the degree to which the design process is formalized. According to Crampton Smith and Tabor (1996) interaction designers go through five steps or activities (Figure 1) in their design process. These activities are almost the same as the steps described in the usability engineering life cycle. The difference is that the interaction designer moves through the steps in an informal manner, whereas the usability engineering process is highly formalized. Another difference is that the interaction designer usually does analysis and evaluation and uses the results in producing a design, while the usability engineer documents the results of analyses and evaluations and hands them over to somebody else.

Which approach is the best – the engineering approach or the design approach, or both? In our research, we have found that combining parts of the formalism in usability engineering with the power of design is one way of promoting usability and user-centered systems design – usability design. In that sense, the usability design process (described later in this article) resembles the process described by Crampton Smith and Tabor (1996). The real challenge in user-centered software development is to maintain a certain degree of formalism, and yet provide enough flexibility and space for the creative and 'unruly' nature of design.

3. USABILITY ENGINEERING – NOT ENOUGH

Many of the methods, tools and techniques from the HCI field fit the usability engineering label (Nielsen 1993, Rosson and Carroll 2002, Faulkner 2000). Usability engineering, as defined by e.g. Preece *et al.* (1994, p. 722), focuses on metrics for measuring usability:

'an approach to system design in which levels of usability are specified and defined quantitatively in advance, and the system is engineered towards these measures, which are known as metrics.'

The emphasis on usability metrics places too much focus on analysis and evaluation, and not enough focus on the actual design process. For instance, recent books on usability engineering contain few pages about design and few examples of design. Usability Engineering by Xristine Faulkner (2000) contains only a few pictures of user interfaces, but a large number of pages with questionnaires, heuristics, methods for analyzing users and evaluation methods. About ten pages out of approximately 230 are dedicated to design (Faulkner 2000: Section 4.7 – Strategies for representing design). About 80 pages describe usability metrics, usability evaluations, design heuristics and expert evaluations. Another 60 pages describe user analysis techniques. We believe that the focus must be shifted to design. User analyses, usability requirements and evaluation results are important in that they provide necessary input to the design activities, but you can only design your way to usability.

Another problem with usability engineering is that it comprises a large range of techniques to analyze users, specify usability goals, evaluate designs, etc – but it does not address the whole development process. Usability engineering techniques are often added on to the software-development process as single activities. The descriptions and applications of the techniques rarely address the problem of how they fit into the overall development process, e.g. what input data are needed from other activities in the process and how the results fit into the process. They rarely define the role being responsible for the usability activity or the deliverables produced in it. We argue that integrating usability into software development requires a process perspective and that the usability professional is involved throughout the entire development process. 'UCD

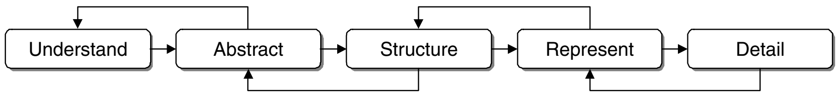


Figure 1. The five activities of interaction design according to (Crampton Smith and Tabor 1996): Understand – what is going on; Abstract – what are the main parts; Structure – how do the parts connect; Represent – how can the structure be represented; Detail – what attributes to use. All of these activities are connected with iterative feedback loops

professionals who focus on doing ''studies'' as opposed to generating designs and products, will always be perceived as peripheral.' (Siegel and Dray 2003, p 20). Mayhew describes usability engineering from a life cycle, process-oriented perspective (Mayhew 1999). We believe that Mayhew succeeds in framing usability activities into a process, but fails to address the design activities. Mayhew states that 'User interface design is key' (1999, p. 9 and 17). Yet, analysis, usability goals and evaluations remain the main focus of the usability engineering life cycle.

We do not believe that there is anything fundamentally wrong with usability engineering techniques, but argue that the integration of usability must be moved one step further, from being separate activities added on to the process to being a natural part of it. We also believe it essential to place the main focus on the design activity/phase. We have, therefore, defined the usability design process – integrating usability into the process and shifting the focus to design within a framework of user-centered software development.

4. ADDRESSING USABILITY IN SOFTWARE DEVELOPMENT

Over the years, there have been various trends regarding usability issues in software development and how to bridge the gap between software engineers and usability professionals (or users or designers or managers). Below, we list and discuss some of the most recent ones:

- *Demand for cost-justification and calculation of return on investments (ROI) for usability efforts*. Cost-justification aims to assess the costs and potential benefits of usability activities. Bias and Mayhew (1994) predicted, in their book on cost-justifying usability, that the trend would be over in five years, but it still seems to be on the agenda (Donahue 2001, Lindegaard 2002, Siegel 2003). Unfortunately, most examples of cost-justification are either hypothetical or describe limited success stories (Nielsen 1993). Currently, times are rough for the IT industry and IT managers are looking for ways of cutting costs. Given these circumstances, we believe that there is a risk that arguments based on costs and potential benefits may be held against usability since 'it requires an up-front investment tied to uncertain returns' (McCoy 2002, p. 285).
- *Promotion of 'quick and dirty' or discount usability engineering methods*. These methods are often marketed as not incurring any extra costs and above all not interfering with existing software development approaches. We believe that the success of such methods is partly due to a mistaken belief that involving users in software development is expensive and of little value. We even catch ourselves saying that it is better do something 'quick and dirty' about usability than nothing at all. However, there are studies that show that discount methods, such as expert evaluations, are of limited value (Molich 2002, Cockton and Woolrych 2002).
- *Usability metrics, nonfunctional requirements, and procurer–supplier relations*. Another problem is that usability is often taken for granted in software development and, therefore, does not get proper attention. The blame is often laid on the client/organization ordering the system (procurer). If they do not include usability as a quality criterion for system acceptance, they will not get a usable system. This has recently led to an increasing emphasis on procurer–supplier relations (Artman 2002) and an even greater focus on usability requirements (Sutcliffe 2002). Usability requirements *are* crucial – if the client does not order a usable system, the developer organization is not likely to spend additional resources on making the system usable. It is, however, far too common that usability requirements are cast aside when time is running out in the development project (McCoy 2002). Thus, usability requirements alone cannot guarantee usable systems.
- *The impact of commercial software-development processes*. Introducing a commercial development process, e.g. the RUP, in an organization often results in a widespread reluctance to do anything but that which is prescribed in the commercial process. Best practices that are well known and widely used in the organization are replaced by new work practices simply because they are part of the commercial package and therefore available and considered a standard. In our earlier research, we have, however, identified problems with how such commercial software-development processes are applied (Gulliksen *et al*. 2003).

- *Finally, there is a trend towards considering the users and user-centered systems design as a source of problems*. Constantine & Lockwood (2002), for example, claim that user-centered design (UCD) is a '... *loose collection of human-factors techniques united under a philosophy of understanding users and involving them in design*'... '*Although helpful, none of these techniques can replace good design. User studies can easily confuse what users want with what they truly need. Rapid iterative prototyping can often be a sloppy substitute for thoughtful and systematic design. Most importantly, usability testing is a relatively inefficient way to find problems you can avoid through proper design*'. (Constantine and Lockwood 2002, p. 43). Their remedy is 'usage-centered engineering' where the emphasis is on the usage, not the users, and on model-driven development. We readily agree with the critique against UCD, but not with the remedy. Model-driven approaches represent a move away from user-centered design, reducing user involvement to that of the users being informants rather than codesigners. User participation is, however, a key success factor for designing for usability (Standish Group 1995, Gould *et al*. 1997, ISO 13407 1999). We argue that software development needs to move towards a user-centered approach rather than away from it. Computer systems (in particular in a work context) must support not only the official rules and version of the work practices but also the particularities in each situation, (Beyer and Holtzblatt 1998, Harris and Henderson 1999) which requires a deep understanding of the context of use. Few development teams have that understanding, and writing requirements documents or creating abstract models is simply not enough to create that kind of understanding. Only the users themselves can provide that.

We believe that the advances described above do not significantly contribute to improving communication between the software engineering community and usability professionals. Instead, we argue that it takes a number of efforts to bridge the gap between software engineering and the HCI field and to put usability on the agenda in software development. Below are some of our conclusions:

- Usability tools, techniques and methods must be integrated in and relate to the software-development process. Usability must be a natural, nonseparable part of the development process.
- Usability professionals must learn more about software development and gain a better understanding of the possibilities and limitations of development tools. Usability professionals often have a noncomputer science background and have limited knowledge and experience of software construction work. Many usability professionals 'fear' design and programming and, consequently, their influence in the project and impact on the process are limited (Siegel and Dray 2003).
- Conversely, software developers must learn more about usability and user-centered systems design (UCSD). Given the relative youth of the HCI field, most software developers have not received any formal education or training in it. It is, therefore, important to teach software developers more about usability and communicate the key principles of UCSD (van der Veer and van Vliet 2003). This would be a lot easier if the software development processes were user-centered and usability was a natural part of software development.
- The software developer community must acknowledge the power of involving users. User involvement is often considered complicated and time-consuming, and something that should be dealt with as quickly and expediently as possible. There is, however, no simple way of involving users and have it done and over with, without too much fuss. User involvement is crucial and the development process must allow for the time it takes.
- Software developers must learn to use design representations of a less formal or model-based nature, such as prototypes and scenarios. Such design representations are concrete and facilitate communication with users on equal terms. Users often have difficulties with understanding use cases, requirements specifications or object models. It is particularly difficult to visualize future work practices and context of use from abstract models.
- The development process must allow enough space and time for interaction design activities that do not benefit from being framed by a formalized process.
- More attention must be paid to the work practices of the multidisciplinary team. The

effective team relies, to a large extent, on effective cooperation, coordination and communication between the individual members. Planning and staffing the team in the best possible way is, therefore, crucial for success.

The above list implies a change in the attitudes to software development. Integrating usability requires a user-centered approach to software development including active and direct user involvement. Our approach, therefore, involves a shift of attitudes towards a user-centered systems design philosophy or paradigm.

We also believe the software-development process could benefit from the ideas and methods that are at the heart of agile approaches (Agile Alliance 2001). Agile processes emphasize the pragmatic use of light, but sufficient rules of project behavior and the use of human and communication-oriented principles (Cockburn 2002). Hence, people are more important than processes and tools. Working software is more important than comprehensive documents and model building. Models and artifacts are only means of communication; consequently prototyping and simple design representations are preferred. Agile developers argue that projects should be communication centric, which implies that effective, face-to-face communication between project members and users is crucial. Agile approaches often rely on direct collaboration with users and customers – preferably, users and developers should share offices during development.

The below definition of and key principles for user-centered systems design embody the attitudes and ideas discussed above.

5. UCSD – DEFINITION AND KEY PRINCIPLES

User-centered systems design is a process focusing on usability throughout the entire development process and further throughout the system life cycle. It is based on the following key principles (see Gulliksen *et al.* 2003 for a more detailed description):

- *User focus*: The goals of the activity, the work domain or context of use, the users' goals, tasks and needs should control the development.
- *Active user involvement*: Representative users should actively participate, early and continuously throughout the entire development process and throughout the system life cycle.
- *Evolutionary systems development*: The systems development should be both iterative and incremental.
- *Simple design representations*: The design must be represented in such ways that it can be easily understood by users and all other stakeholders.
- *Prototyping*: Early and continuously, prototypes should be used to visualize and evaluate ideas and design solutions in cooperation with the users.
- *Evaluate use in context*: Baselined usability goals and design criteria should control the development.
- *Explicit and conscious design activities*: The development process should contain dedicated design activities.
- *A professional attitude*: The development process should be conducted by effective multidisciplinary teams.
- *Usability champion*: Usability experts should be involved from the start of project to the very end.
- *Holistic design*: All aspects that influence the future use situation should be developed in parallel.
- *Process customization*: The UCSD process must be specified, adapted and implemented locally in each organization. Usability cannot be achieved without a user-centered process. There is, however, no one-size-fits-all process.
- *A user-centered attitude must be established*: UCSD requires a user-centered attitude throughout the project team, the development organization and the client organization.

This definition and these key principles have been put together to reflect that UCSD is about changing the attitude among all professionals involved in the software development process. In order to provide necessary support for these professionals, we need a user-centered systems design process – usability design. The usability design process gives equal weight to interaction design, analysis and evaluation, combining interaction design, (Cooper 1999) and usability engineering (Mayhew 1999). We must furthermore specify the usability design process so that it can be related to structured software development processes. UCSD must be brought down to the level of 'boxes and arrows' in order to gain acceptance from the software-development community. This is what the usability design process intends to do.

6. USABILITY DESIGN – A BRIEF INTRODUCTION

Usability design is our attempt to 'put a face' on UCSD in order to get organizations and projects to start adopting parts of the UCSD philosophy. It is a result of extensive research and practical experiences in a number of different organizations and projects (Gulliksen *et al*. 2001, Frisk *et al*. 2001, Gulliksen and Göransson, 2001). One of our main rationales for the concept is that clients (buyers of software development) want a design solution. They are not particularly interested in HCI methods and theories. We have also found that development organizations often find it difficult to apply usability engineering 'by the book', and to fully appreciate its benefits and potential. Usability design is a lightweight UCSD process that is in use in several development projects in practice. We have adopted the term usability design from Gould, Boies and Ukelson (1997) and their principles for designing for usability. We define *usability design* as:

> *a user-centered design approach for developing usable interactive systems, combining usability engineering with interaction design, and emphasizing extensive active user involvement throughout the iterative process.*

The usability design process is based on the key principles for user-centered systems design (Gulliksen *et al*. 2003) described in the introduction. It has its base in usability and HCI, and the 'usability' in usability design is particularly important. It implies the strong standpoint in the definition of usability.[1] (ISO 9241-11 1998) and that the process does not rely solely on the designers' skills. Usability design is, furthermore, inspired by usability engineering, but not synonymous to it. Usability design has adopted concepts from the usability engineering life cycle (Mayhew 1999) but we have simplified and adapted the process to the organizations and projects we have been involved in. A more explicit focus on design has been added. In her book, Mayhew (1999, p. 9 and p. 17) states:

> *'User interface design is key. I present usability engineering tasks in the foreground, with less emphasis on the overall software engineering tasks, to communicate my belief that user requirements and user interface design should drive the overall development process, rather than be driven by it or be incidental to it. After all, the whole point of interactive products is to serve the users, and as far as users are concerned, the user interface is the product.'*

We fully agree with Mayhew. But in our research and practical experiences we have found it necessary to focus even more on the design. As mentioned above, clients in general want solutions (i.e. the design). A successful user-centered systems design process must focus on providing that solution – the design. We have, therefore, combined the usability engineering approach with some of the ideas and principles from interaction design (Cooper 1999).

7. THE USABILITY DESIGN PROCESS

The figure (Figure 2) fits the usability design process into the overall user-centered systems design life cycle (adopted from ISO/TR 18529:2000(E)).

Pre-study and business analysis can be anything from a comprehensive analysis of work procedures, business processes, etc., to a brief statement or vision. *Planning the user-centered systems design process* includes setting up the project with resources,

[1] Usability is defined as 'the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction, in a specified context of use' (ISO 9241–11 1998), Please note that this definition includes the concept of utility or usefulness, often seen as separate from usability.

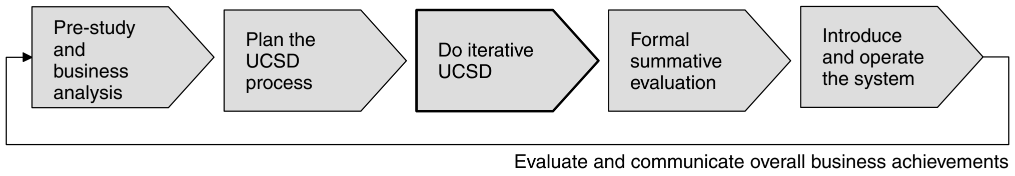

Pre-study and business analysis
Plan the UCSD process
Do iterative UCSD
Formal summative evaluation
Introduce and operate the system
Evaluate and communicate overall business achievements

Figure 2. A suggested life cycle perspective on user-centered systems design

activities, roles, methods, etc. The usability design process approximately matches the box '*Do iterative UCSD*'. The formal summative evaluation covers the usability of the resulting system, as opposed to the formative evaluations used in the usability design process to learn about details in the design (Nielsen 1993, p. 170). *Introduce and operate the system* includes installation, change management, user training, evaluating long-term effects and so forth.

Figure 3 outlines the steps in the development process involving usability design aspects. The principles for designing for usability (Gould *et al.* 1997) (see the upper right corner of the figure) are included to lay the foundations of the usability design process. The process is iterative and should be integrated in the overall development process. The process contains activities that can be carried out by means of various methods. The methods are selected in an earlier stage (i.e. *Planning the UCSD process*) and we will not discuss here what methods are applicable and suitable. This framework serves the dual purpose of providing an artifact for communication and a process guiding the UCSD process. The usability design process is not a complete software development process, but in the future, we intend to frame usability design into a more comprehensive development process.

The process can be divided into three main phases: *requirements analysis*, *growing software with iterative design* and *deployment*. In the following sections, we discuss each phase and illustrate them with examples taken from some of the projects where we have applied the process. The examples are from 'real life', taken from a consultant company involved in bespoke software development for various clients.

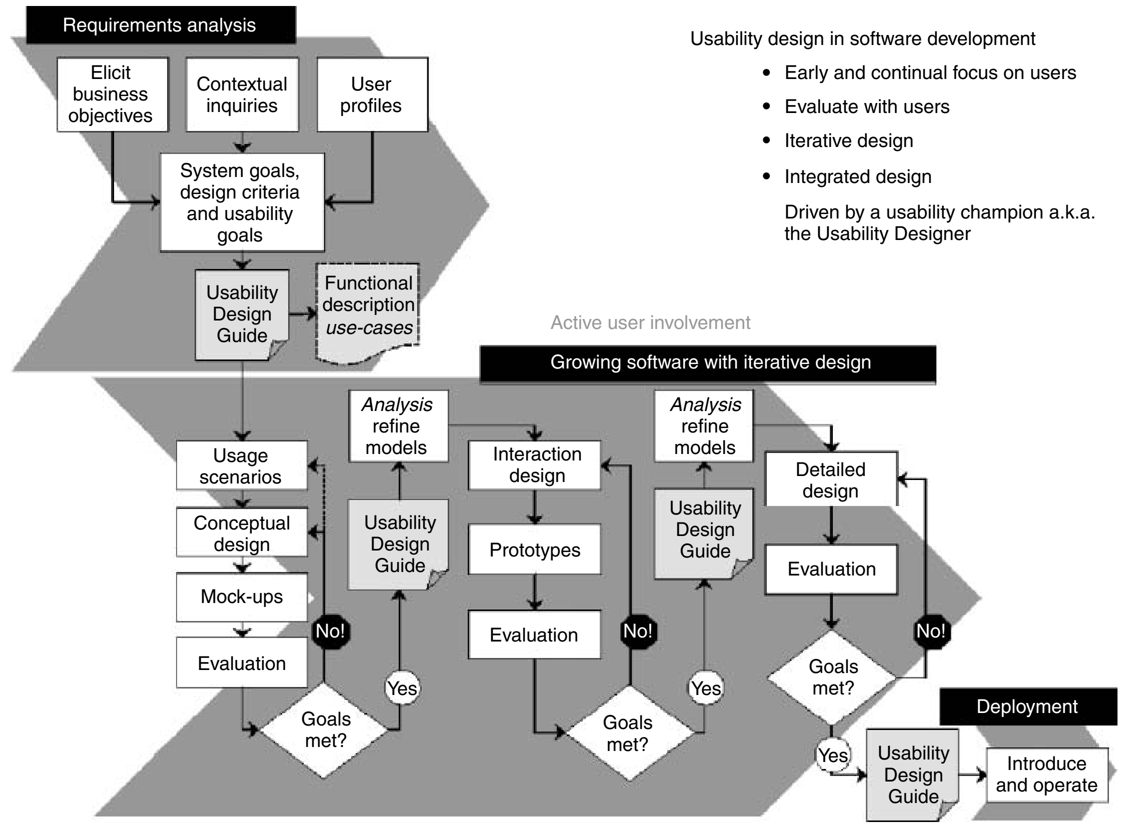
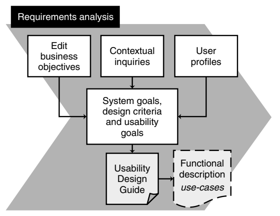


Figure 3. The usability design process

8. REQUIREMENTS ANALYSIS



The *requirements phase* is in focus initially. Later on in the process, it is 'revisited' whenever major changes occur in the functionality, work tasks, user groups, etc. This is important to remember since the graph of the process may give the impression that the requirements are never changed once you leave the requirements phase (reverting to the waterfall process). On the contrary, requirements evolve continuously during the design process. As long as you baseline the fundamentals of the system, you will have enough information to guide the design.

In this phase, we focus on understanding the business objectives; the targeted user groups, their work tasks and needs; and establishing system goals, design goals and usability goals.

The level of detail of *business objectives* differs from organization to organization. Some organizations have a complete set of business processes at hand and can point out precisely what kind of support they need to fulfill their business objectives. Others may be very vague about their business objectives and have a vision rather than goals. In such cases, the objectives should be elicited and made visible before moving on in the process.

The project must have a clear understanding of the future users of the system, their work tasks and needs. The individual user groups have different characteristics and needs, which must be taken into account when designing the system. There are several ways to learn about your users. In our projects, we use *user profiles* to analyze the users with respect to their background, skills, abilities, etc. We also conduct field studies – *contextual inquiries* (Beyer and Holtzblatt 1998, Mayhew 1999) – to capture what users do at work, their work tasks, goals, work environment, etc., and to a large extent their future needs. We believe that field studies or contextual studies are essential for capturing tacit knowledge, informal work practices and information that is 'hidden' in the context. We have also used the concepts of personas and goal-directed design, as described by Cooper (1999) to complement the user profiles.

The *goals for the system, design and usability* are important for baselining the usability requirements and the design process. As usability goals are not easy to identify and use, we often combine them with design criteria to drive the design process (Borälv and Göransson, 1997). Usability goals can be used for acceptance testing or to identify trends over time, whereas design criteria are high-level criteria derived from the usability goals, providing guidance for design decisions.

Currently, *use cases* are widely used in development organizations for capturing requirements and driving the development process (Jacobson *et al.* 1999). They provide a *functional description* of the system. The usability design process fits into use case–driven processes (e.g. Rational Unified Process) complementing them with a usability focus.

9. USABILITY DESIGN GUIDE

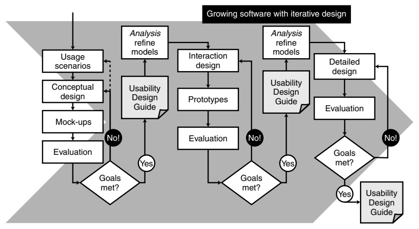
The results from the requirements phase, and the other phases, can be documented in a Usability Design Guide. The Usability Design Guide is specific to the system under development. It contains the output from the usability design process including user categories, personas (if used), work task analyses, usability goals and design criteria, scenarios, conceptual design and detailed design, design decisions, etc. Below is a 'template' table of contents for the Usability Design Guide.

- *Customizing the usability design process for the project*: Contains a detailed description of the activities and methods in the project. Details the integration with other parts of the process framework, as well as roles and artifacts.
- *Plan for user participation*: Contains details about user participation – how, when, to what extent, who, etc.
- *Overview of the system – goals and functionality*: Contains a high-level description of the main goals of the system and an outline of the functionality that will be provided.
- *User profiles and/or personas*: Describes the different user categories with respect to characteristics, backgrounds, special needs, goals, etc.
- *Contextual task analysis*: Contains the results from field studies, user interviews, task analysis, etc.

- *Platform capabilities and constraints*: Describes technical constraints, their effects on usability, the use of platform specific style guides, etc. May also cover aspects such as legal constraints.
- *Usability goals*: Lists the usability goals and specifies their use in the evaluations.
- *Design decisions and criteria*: Outlines the design criteria that 'drive' and shape the design. Should also cover design decisions and their rationales.
- *Usage scenarios*: Contains scenarios describing how the system will be used by users in context.
- *Conceptual design*: Contains paper sketches, explaining text, etc. outlining the user interface architecture and structure – the 'real estate' of the screen. May also cover branding aspects or other aspects of the design that are important on the conceptual level.
- *Interaction design, navigation and information structure*: Contains sketches, mock-ups and pictures (and even screenshots) specifying the interaction, navigation, information structures and the dynamics of the user interface.
- *Detailed design*: Contains a detailed specification of all design components, down to virtually every pixel of the screen; windows, data fields, menus, buttons, graphics, colors, typefaces, etc.
- *Design artifacts*: Contains a collection of pictures of the final user interface accompanied by explaining text. Often combined with running prototypes.
- *Feedback and evaluations*: Contains all the information and data gathered in evaluations and the user feedback given during the development.

The contents of the Usability Design Guide must be adapted to the needs of each individual project or organization so that it fits into the overall project documentation and serves its purpose without adding unnecessary overhead. The level of detail and scope varies a great deal, depending on, for instance, the software-development process. In an agile process, the Usability Design Guide may contain no more than personas, user stories and some screen dumps to specify the user interface. In a more formalized process, the Usability Design Guide may be divided into several different documents or artifacts containing a high level of detail.

10. GROWING SOFTWARE WITH ITERATIVE DESIGN



The growing software phase contains three major loops: *conceptual design*, *interaction design* and *detailed design*. The loops are iterative, comprising design, evaluation, analysis, redesign, evaluation and so on. The design and the system 'grow' as we go through the loops. Over time the design becomes increasingly detailed. We use usage scenarios to drive the exploration of design solutions. As always, users participate in these activities. While a process cannot, in itself, guarantee that development will involve users, it can encourage a user focus and provide opportunities for user involvement and feedback.

The conceptual design is a high-level baseline of the overall user interface structure. Usually, it is based on some sort of metaphor, e.g. the workspace metaphor (Borälv and Göransson, 1997). The initial conceptual design is typically done as simple sketches.

Figure 4 shows examples of early conceptual design sketches for a system. There are six variations on a theme. In Figure 5, one conceptual design is selected and further outlined.

As we move on to the interaction design, we outline the structure of the interaction and navigation. Figure 6 illustrates a conceptual navigation structure for a system.

In Figure 7, chunks of information have been identified and placed on the screen, together with the functionality that comes with the information. Navigation paths are identified and outlined.

Detailed design takes us down to the individual parts of the screen and to data fields, input fields, menus, buttons, etc. Figure 8 shows a detailed part of the user interface with all information and functions in place.

Figure 9 contains examples of detailed data entry fields used throughout the system.

Note that there is no strict sequence between the three loops, but we recommend that you start

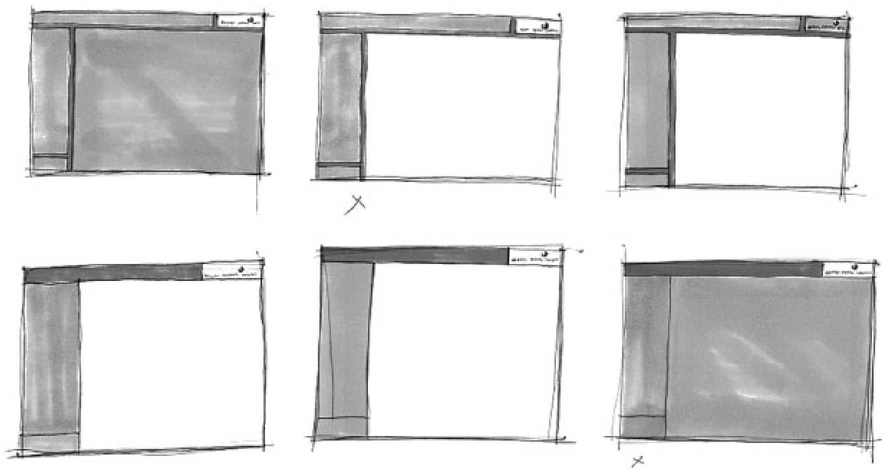


Figure 4. Examples of different, paper-bound, conceptual design suggestions for a system

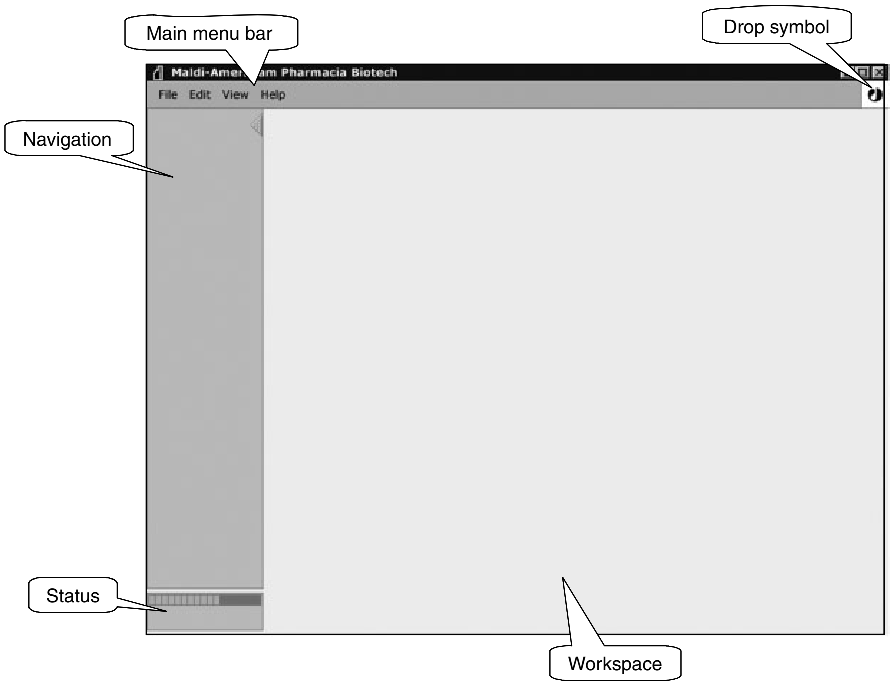


Figure 5. The conceptual design; annotated and outlined

with the conceptual design before going into more details. In an incremental development process, the design phase delivers appropriate increments of the user interface and interaction to the rest of the process. In a nonincremental development process, the results of the design loops are part of the specification of the system.

Each loop contains activities where the design solution is evaluated together with users. An iteration must include at least a proper analysis of

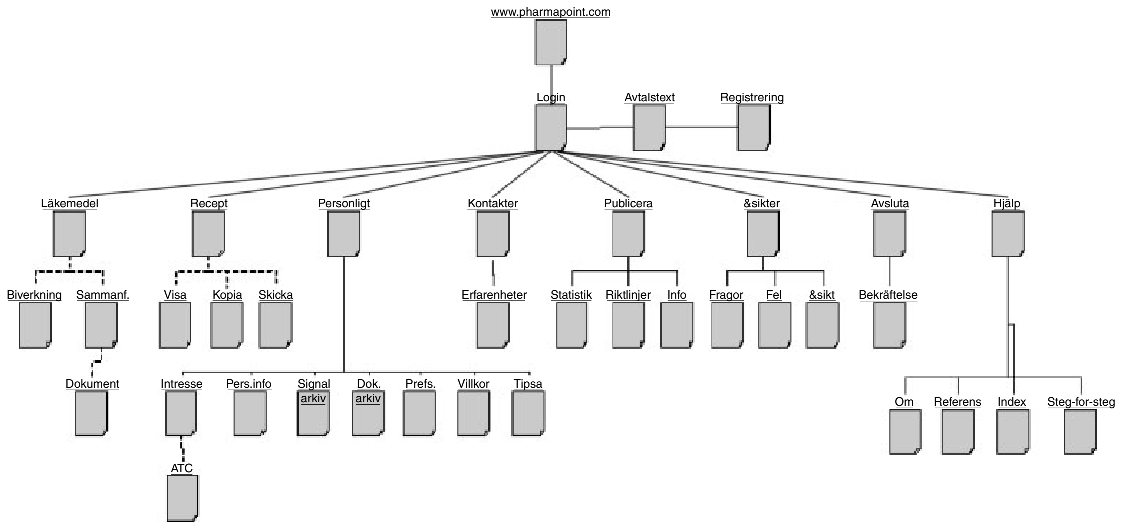
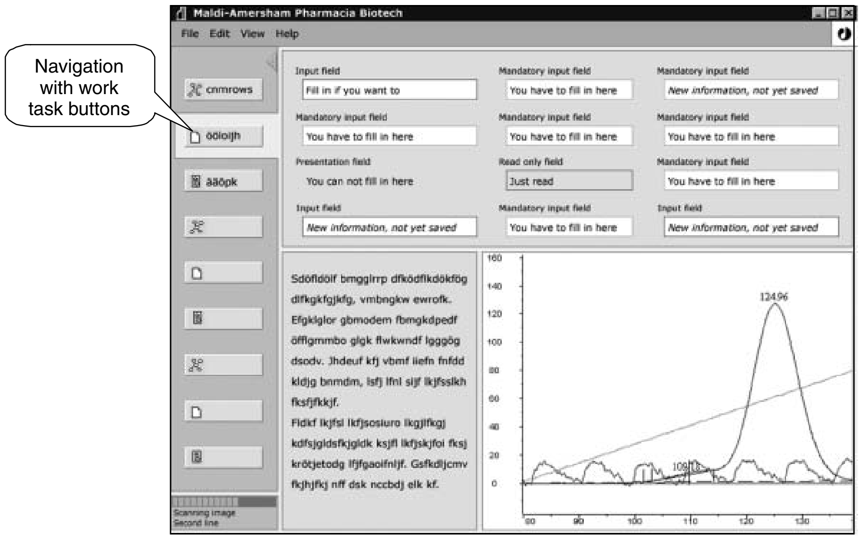


Figure 6. Example of a navigation structure



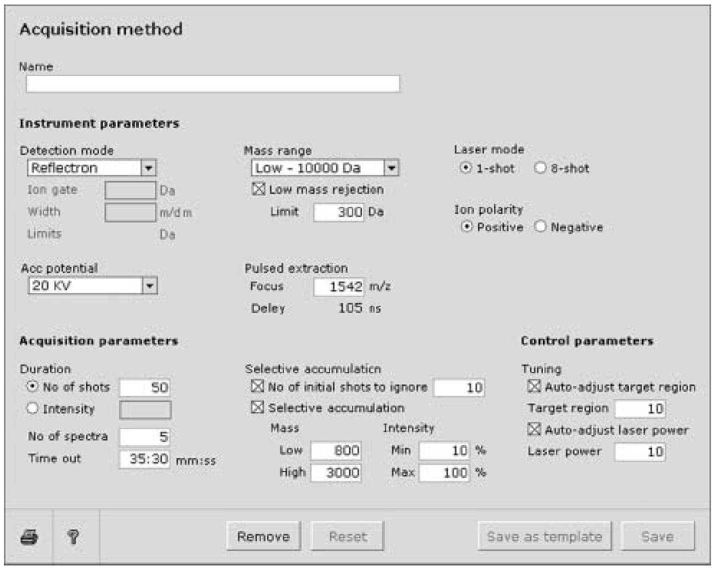
Input field: Fill in if you want to
Mandatory input field: You have to fill in here
Mandatory input field: *New information, not yet saved*
Mandatory input field: You have to fill in here
Mandatory input field: You have to fill in here
Mandatory input field: You have to fill in here
Presentation field: You can not fill in here
Read only field: Just read
Mandatory input field: You have to fill in here
Input field: *New information, not yet saved*
Mandatory input field: You have to fill in here
Input field: *New information, not yet saved*

Sdöfldölf bmgglrrp dfködflkdökfög difkgkfgjkfg, vmbngkw ewrofk. Efgklglor gbmodem fbmgkdpedf öfflgmmbo glgk flwkwndf lggög dsodv. Jhdeuf kfj vbmf iiefn fnfdd kldjg bnmdm, lsfj lfnl sijf lkjfsslkh fksfjfkkjf.
Fldkf lkjfsl lkfjsosiuro lkgjlfkgj kdfsjgldsfkjgldk ksjfl lkfjskjfoi fksj krötjetodg lfjfgaoifnljf. Gsfkdljcmv fkjhjfkj nff dsk nccbdj elk kf.

Figure 7. Prototype showing concepts from the interaction design

the user requirements and the context of use; a prototype design phase; and *a documented evaluation of the usability of the prototype* suggesting modifications in the design. The design solution should be evaluated against usability goals and proper actions should be taken in the next iteration. However, it is rarely possible to iterate until all the goals are met. Aspects such as deadlines, budget constraints and 'good enough' are equally important.

What evaluation methods you use depends on the purpose, there are plenty to choose from (Mayhew 1999, Faulkner 2000, Nielsen 1993, Nielsen and Mack 1994). The important thing is to involve real users and not only the ones already involved in the project. There



Acquisition method

Name:

Instrument parameters

Detection mode: Reflectron
Ion gate: Da
Width: m/d m
Limits: Da

Mass range: Low - 10000 Da
☒ Low mass rejection
Limit: 300 Da

Laser mode: [✓ 1-shot] 8-shot

Ion polarity: [✓ Positive] Negative

Acc potential: 20 KV

Pulsed extraction
Focus: 1542 m/z
Deley: 105 ns

Acquisition parameters

Duration
[✓ No of shots]: 50
Intensity:
No of spectra: 5
Time out: 35:30 mm:ss

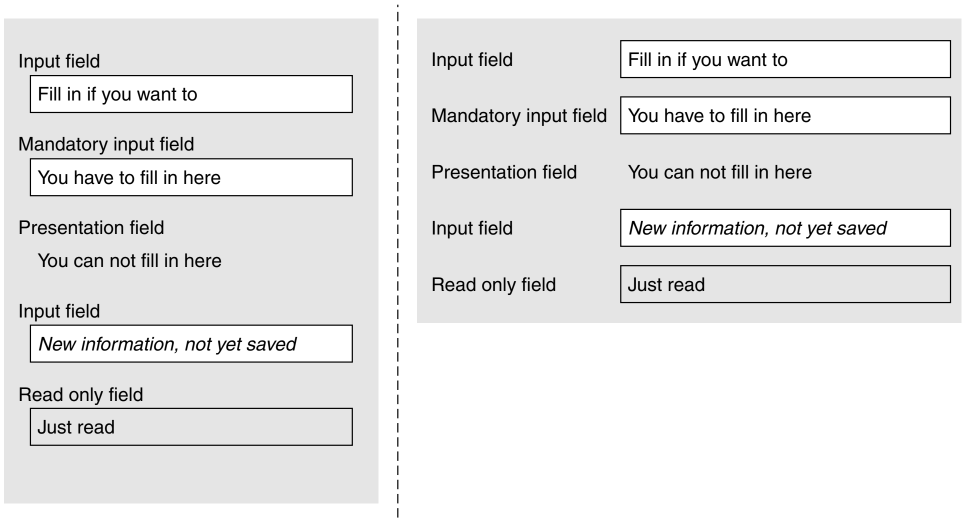
Selective accumulation
☒ No of initial shots to ignore: 10
☒ Selective accumulation

Mass		Intensity	
Low	800	Min	10 %
High	3000	Max	100 %

Control parameters

Tuning
☒ Auto-adjust target region
Target region: 10
☒ Auto-adjust laser power
Laser power: 10

Figure 8. A detailed part of the system



Input field
Fill in if you want to

Mandatory input field
You have to fill in here

Presentation field
You can not fill in here

Input field
New information, not yet saved

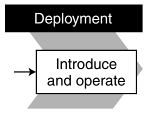
Read only field
Just read

Input field	Fill in if you want to
Mandatory input field	You have to fill in here
Presentation field	You can not fill in here
Input field	*New information, not yet saved*
Read only field	Just read

Figure 9. Example of design details: making up the standard interaction elements for the system

is little point in using sophisticated evaluation methods producing quantitative data early on in the project. We typically use informal evaluation methods on the conceptual design. Such methods are good for producing insights into the pros and cons of a particular design and to set a direction for the design, but they do not produce numbers. We agree with Nielsen (on-line Alertbox, February 18, 2001: http://www.useit.com/alertbox/20010218.html) that the most effective usability evaluation technique are frequent and simple tests with users thinking out loud while performing tasks in the system.

11. DEPLOYMENT



The *Deployment* phase, i.e. introducing and operating the system, differs from project to project. It can be anything from creating a commercially available software package including on-line help, user manuals, user training, etc., to simply distributing a CD within an organization. In our experience, the deployment phase is critical for the success of the system. Far too often, well-designed and well-conceived software is simply dumped on the users, with no on-line help, manuals or proper training. The deployment phase involves different people and includes different activities depending on the type of organization. The most important factor is that deployment must be planned up-front in the project and not be put off till later – later always seems to be too late. Delivering the system in increments makes it possible to introduce the system with fewer resources and more opportunities to fine-tune the process and the system, increment by increment.

The process does not end with the deployment; it is rather the starting point for the next cycle in the overall life cycle of the system.

12. SUMMING UP THE USABILITY DESIGN PROCESS

Ideally, the usability design process is conducted by a team led by the usability designer (Göransson and Sandbäck 1999). Other specialized competencies are needed as well: field study specialists, interaction designers, graphic designers, usability evaluation specialists, developers for implementing prototypes, etc. The number of people and competencies involved differ from project to project. By defining a process, it is possible to specify critical activities regardless of how they are conducted and by whom.

As mentioned above, the usability design process is *not* a complete development process. It must be used within the context of such a process. Nevertheless, it is the usability design process that underlies the user-centered philosophy that should inspire and be reflected in the whole development process. The usability design process must be tailored for every organization (Gulliksen *et al.* 2003). It does not explicitly detail all parts of the UCSD process, for instance, when and how to involve users, but conveys the underlying message that users must be present and active throughout the whole process. There are, of course, activities and artifacts that are crucial in a user-centered development project. Including such details in the process would complicate the picture too much for our purpose, which is to convince people that user-centered systems design is something that they should go ahead with and actually use in their development projects.

13. EXAMPLE: USABILITY DESIGN AS A DISCIPLINE IN RATIONAL UNIFIED PROCESS

The *usability design discipline* (UDd) (Göransson *et al.* 2003) is an example of how usability design can be integrated into a commercial systems development process, the Rational Unified Process (Kruchten 1998). We choose the RUP because it is one of the most widely used software-development processes in Sweden[2]. We believe it is necessary to relate usability design to the RUP in order to get acceptance from software developers in practice.

The UDd is based on our previous research and extensive practical experience as usability consultants in large software-development projects. The UDd was developed as a plug-in to the RUP and does not include any advice on how to modify the other disciplines to better include and make use of the artifacts and results produced in UDd (Figure 10).

The UDd complements the RUP. It promotes and facilitates usability work within the software-development process by means of a user-centered approach. Most of the usability-related activities are performed during the inception phase and the early phases of elaboration. During the construction and transition phases, UDd primarily includes monitoring and ad hoc design decisions. GUI coding is not part of the UDd.

[2] The RUP has become more or less the *de facto* standard for software development in Sweden. There are currently 6000 license owners in Sweden and almost all the development organizations we are studying use RUP. According to Rational (Rational Software originally developed the RUP. They have since been acquired by IBM), the RUP has been used in more than 10 000 customer projects worldwide and it is part of the computer science curriculum in hundreds of universities.

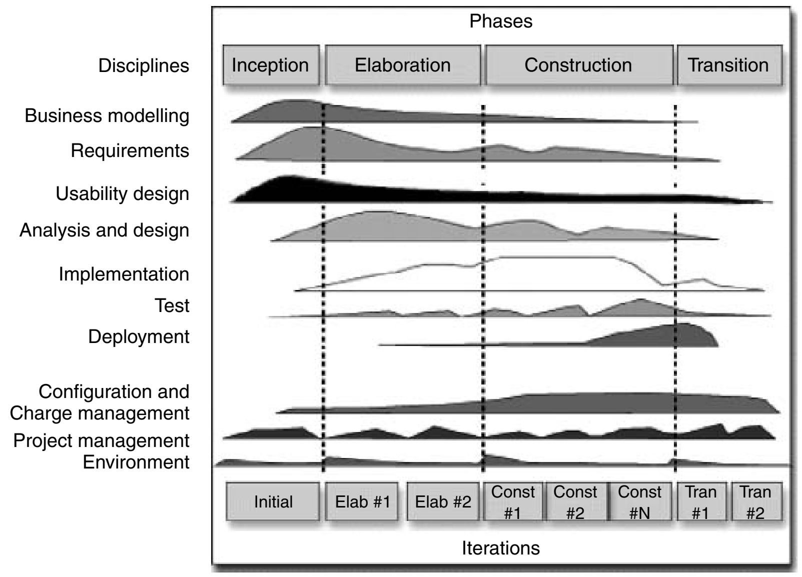


Figure 10. The usability design discipline as part of the overall architecture of the RUP, a modified version of the process description overview in the RUP

The UDd workflow (Figure 11) is an iterative process, where the design solutions are repeatedly evaluated and modified in accordance with the evaluation results.

The main activities in the UDd are:

- *Usability design plan*: Involves detailed planning of the user-centered activities and user involvement. The plan is refined in each iteration.
- *Conduct user studies*: Interviews, observations, workshops, etc – to understand the potential users of the system, their needs and the context of use. Specify the goals for the system, the design criteria and usability goals.
- *Perform competitor analysis*: To get inspiration from similar state-of-the-art systems or products.
- *Conceptual design*: Describes the overall structure of the user interface. Use brainstorming, usage scenarios, sketches and mock-ups to illustrate potential high-level designs solutions.
- *Interaction design*: Outlines details in the conceptual design illustrating potential user interaction (navigation, information and functionality), simulating the real system.
- *Detailed design*: Includes fine-tuning all the design details in the GUI
- *Develop user assistance*: A parallel design activity focusing on on-line help systems, manuals and user training material. Also covers new work procedures.
- *Monitor usability work*: Handling late change requests and ad hoc decisions in the construction phase.
- *Usability evaluation*: Of design solutions against the usability goals. Evaluations should be performed on all design suggestions and solutions, from preliminary sketches to fully interactive prototypes and the final system.

Each of the main activities has been detailed (Figure 12), describing the main activities, the roles involved and the artifacts produced.

For more details on the UDd, refer to (Göransson *et al*. 2003).

14. LESSONS LEARNED AND DISCUSSION

The usability designer role (Göransson and Sandbäck 1999) was the starting point of our work with the usability design process. The process was modeled on the work practices of the usability designer and is an attempt to compile the best practices of that role. There are still traces of

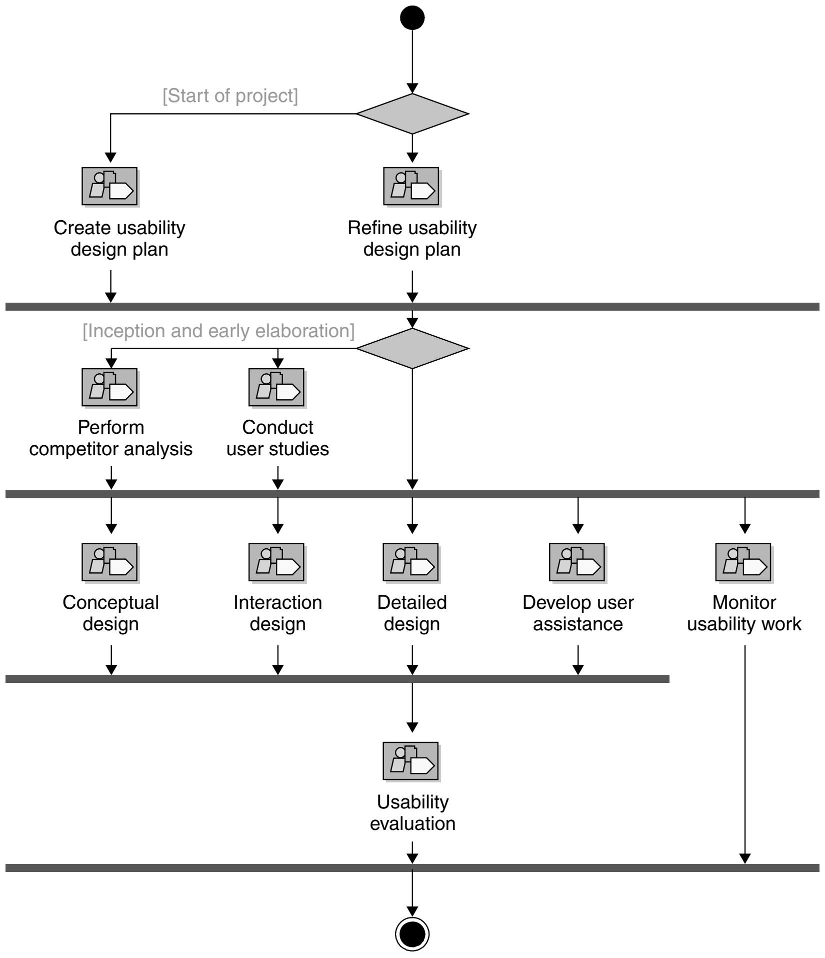


Figure 11. The core workflow of the usability design discipline

the usability designer role in the process, which makes it difficult to apply the process without the role. We have, therefore, tested the process only in projects or organizations that have the usability designer role. Even so, the usability design process is not just for the usability designer, it has implications for the development project as a whole.

The usability design process is based on knowledge and insights gained from research efforts, theoretical studies, practical experiments and our own practical experiences from several development projects and organizations. Currently, the usability design process is used in a couple of development organizations, but with different 'flavors'. One example is Enea Redina[3], a consultant company. They use the process to some extent in all their development projects, but the process never looks the same from one project to another – simply because the development process always has to be adapted to the particularities of the situation, the project and the organization. The usability design plug-in for the RUP, described above, is currently being made part of

[3] Enea Redina is a Swedish IT consultant company practicing UCSD. More information can be found at http://www.redina.se/.

the RUP development case in one large in-house development organization (in a Swedish national authority).

In our work with applying and evaluating the process, we have encountered a number of advantages, but also some drawbacks with using the usability design (UD) process. Below we list and discuss some of them.

Positive	Negative
Is a process – the UD process details the key activities that constitute user-centered systems design on a 'boxes-and-arrows' level.	Does not comprise the whole process – the UD process must be integrated in an existing software process – it cannot stand alone.
An effective communicator – the UD process effectively communicates how to structure usability work in a process, and how to include usability in business agreements, etc.	Can be excluded – the UD process is sometimes perceived as one unit that is either in or out. Thus, it can be excluded in the software-development process.
Makes usability visible – the UD process makes usability visible in projects and in development frameworks, such as the RUP.	Makes usability somebody else's problem – the UD process may provide an excuse for the project members to place the whole responsibility for usability on the UD process and the usability designer and to stop paying attention to it altogether.
Learning tool – the project members can learn about usability by using the UD process or taking part in the activities, or just by using the artifacts and results produced in the process.	

The usability design process was defined to add a process perspective to usability work in software development. We wanted to provide more detail than generic frameworks do, e.g. ISO 13407, but to be less rigorous than, e.g. the RUP. Most people involved in software development would, when asked, say that they work in accordance with some predefined software-development process. However, in one of our previous studies, we found mismatches between what the software developers said they do at work and what they actually did. On an overall process level, the developers complied with the software-development process explicitly used in the organization, but on a day-to-day basis they used implicit work procedures that fitted into the overall process. (Boivie *et al.* 2003). Our conclusion is that formal software-development processes mainly serve the purpose of communicating the intended work practices. Detailed and rigorous guidelines about what to do, when and how, become rather meaningless in that perspective. People do what they have always done, anyway. Thus, there is little point in providing detailed guidelines and rules of behavior.

Hence, the communicative and explanatory aspects of the usability design process are very important. The process can be, and is, used to communicate what aspects, activities, etc, are important for shifting the focus of the software-development process towards usability and user-centered design. The communicative power of the process also makes it useful as a sales argument in a client–supplier relation.

As discussed above, we believe that usability must be tightly integrated into the software-development process. Usability techniques, methods and tools that are not a natural part of the process are easy to leave out when the project is running out of time or money. Having a whole process makes it more difficult to abandon usability when the deadline is approaching. The process makes sure that much of the usability efforts are done up-front in the project. Nevertheless, this does not entirely eliminate the risk of usability being abandoned in the development process as the whole usability design process can be left out. In order for the process to work, the organization or project must be very explicit about using it and integrating it into their software development process. This includes adapting, or tailoring all parts of the overall development process to make sure that it

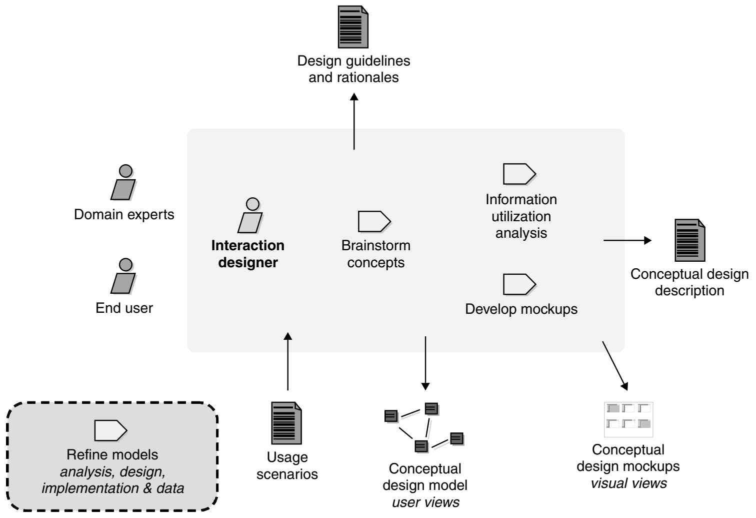

Design guidelines and rationales
Domain experts
End user
Interaction designer
Brainstorm concepts
Information utilization analysis
Develop mockups
Conceptual design description
Refine models analysis, design, implementation & data
Usage scenarios
Conceptual design model user views
Conceptual design mockups visual views

Figure 12. Workflow detail: conceptual design

supports a user-centered approach where usability is not just the concern of usability professionals but the concern of everyone involved in the process.

So far, we have only tested the process in contract development or in-house development in work-related contexts, i.e. bespoke software for professional users. Would it be applicable in other development contexts, for instance, web site design? We have not tested it, but would like to argue that the basics are the same – use a user-centered systems design philosophy to design usable software, whether on the web or not. We see no reasons that the usability design process and the usability designer would not fit into web projects as well.

In parallel with our work on the process, we have continued experimenting with the usability designer role (Göransson and Sandbäck 1999). Both the role and the process are intended as tools for integrating usability aspects and a user-centered approach into the software-development process. Making your software development user-centered requires major changes in attitudes, processes and the organization (Gulliksen *et al*. 2003). We see the usability designer as the first step in such a 'transformation'. The usability designer is, in some senses, a usability champion who promotes usability in the projects and the organization. He/she also participates in project work, conducting usability activities in accordance with the key principles for UCSD (described above). When the role has been established, the organization can move on to adopting the usability design process as part of their software-development process. This is the next step in shifting the whole organization towards a user-centered approach.

To sum it up, the overall purpose of the usability design process is to provide a tool for integrating usability aspects into the software-development process. The process has been applied in different projects and organizations, and we are currently evaluating both the process and the role. We are still a long way away from usability and user-centered design being a natural part of software development. But, we believe that the process is the first important step in changing the attitudes towards UCSD in software development.

15. FUTURE WORK

Current and future work includes evaluations of the usability designer role (Boivie *et al*. 2004) and of the application of the usability design process in projects in practice (Göransson, 2004).

The next step is to integrate usability design into the framework of a complete user-centered development life cycle. We have outlined such a process in Figure 13.

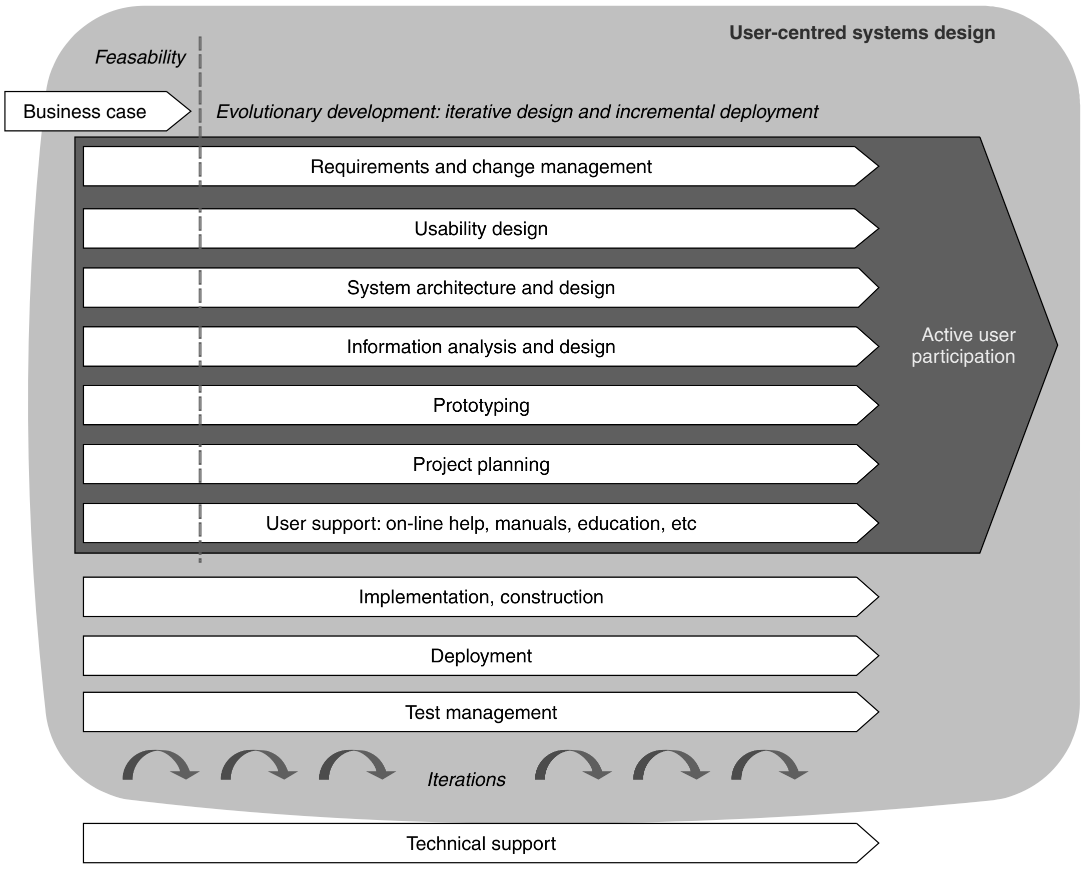

User-centred systems design
Feasability
Business case
Evolutionary development: iterative design and incremental deployment
Requirements and change management
Usability design
System architecture and design
Information analysis and design
Prototyping
Project planning
User support: on-line help, manuals, education, etc
Active user participation
Implementation, construction
Deployment
Test management
Iterations
Technical support

Figure 13. Draft framework for a complete user-centered systems design process showing workflows from left to right. The process is both iterative and incremental

The process is a modification and an extension of the results we obtained when modeling the software development process at the Swedish National Tax Board (Gulliksen and Göransson 2001). It is based on a user-centered systems design philosophy and covers all parts of the software development process. We intend to detail the process and test it in future development projects.

ACKNOWLEDGEMENTS

This article reports on work that was performed with financial support from the National Agency for Innovation Research (VINNOVA) and the Council for Work Life and Social Science Research (FAS). The usability design process was developed in cooperation with Enea Redina AB. All cooperation from those who have influenced this work is greatly appreciated.

REFERENCES

Agile Alliance. 2001. *Manifesto for Agile Software Development*. Available at http://www.agilealliance.org.

Artman H. 2002. Procurer usability requirements: negotiations in contract development. In *NordiCHI 2002. Proceedings of the Second Nordic Conference on Human-Computer Interaction*, Berthelsen O, Bødker S, Kuutti K (eds). ACM Press, New York, USA.

Beyer H, Holtzblatt K. 1998. *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann: San Francisco, CA.

Bias R, Mayhew D (ed.). 1994. *Cost-Justifying Usability*. Academic Press.

Boivie I, Göransson B, Gulliksen J. 2004. The lonesome cowboy – a study of the usability designer role in systems development. Submitted to *Interacting with Computers*.

Boivie I, Gulliksen J, Göransson B. 2003. It's all in a days work of a software engineer. In *Human-Computer Interaction Part 1, Proceeding of HCI International 2003*, Jacko J, Stephanidis C (eds). Lawrence Erlbaum Inc.

Boivie I, Åborg C, Persson J, Löfberg M. 2003. Why usability gets lost or usability in in-house software development. *Interacting with Computers* **15**(5): 623–639.

Borälv E, Göransson B. 1997. A teleradiology system design case. In *Designing Interactive Systems: Processes, Practices, Methods and Techniques*, Van der Veer G, Henderson A, Coles S (eds). ACM Press, New York, USA, 27–30; *DIS '97 Conference Proceedings*, Amsterdam, August 18–20 1997.

Carroll J (ed.) 2000. *Making Use: Scenario-Based Design of Human-Computer Interactions*. MIT Press, ISBN 0262032791.

Cockburn A. 2002. *Agile Software Development*. Addison-Wesley: Boston, MA.

Cockton G, Woolrych A. 2002. Sale must end: should discount methods be cleared off HCI's shelves? *Interactions*, Vol. IX. 5. ACM.

Constantine LL, Lockwood LAD. 1999. *Software for Use – A Practical Guide to the Models and Methods of Usage-Centered Design*. Addison-Wesley Longman, Inc: Reading, MA.

Constantine LL, Lockwood LAD. 2002. User-centered engineering for web applications. *IEEE Software* **19**(2): 42–50.

Cooper A. 1999. *The Inmates are Running the Asylum: Why High-Tech Products Drive us Crazy and How to Restore the Sanity*. SAMS: Indianapolis, IN, ISBN-0-672-31649-8.

Crampton Smith G, Tabor P. 1996. The role of the artist-designer. In *Bringing Design to Software*, Winograd T (ed.). ACM Press: New York.

Donahue GM. 2001. Usability and the bottom line. *IEEE Software* **18**(1): 31–37.

Fallman D. 2003. Design-oriented human-computer interaction. *Proceedings of CHI* **5**(1): 225–232.

Faulkner X. 2000. *Usability Engineering*. Palgrave: New York, ISBN 0-333-77321-7.

Fowler M. 1997. *UML Distilled. Applying the Standard Object Modeling Language*. Addison-Wesley Longman Inc: Reading, MA.

Frisk A, Göransson B, Sandbäck T, Thomasson V. 2001. The design of a smart card-based home-help system. In *Proceedings of INTERACT 2001*, Hirose M (ed.). IOS Press.

Gamma E, Helm R, Johnson R, Vlissides J. 1995. *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison-Wesley: Boston, MA.

Göransson B. 2004. User-Centred Systems Design: Designing Usable Interactive Systems in Practice. PhD thesis. Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology, ISSN 1104-232X; 981 Uppsala: Acta Universitatis Upsaliensis.

Göransson B, Sandbäck T. 1999. Usability designers improve the user-centred design process. *Proceedings of INTERACT '99*, Edinburgh, UK.

Göransson B, Lif M, Gulliksen J. 2003. Usability design – extending rational unified process with a new discipline. In *Interactive Systems, Design, Specification and Verification*, Jorge J, Nunes N, Cunha J (eds). Springer-Verlag: Berlin, Heidelberg, New York, 316–330; *10th International Workshop, DSV-IS 2003*, Funchal, Madeira Island, Portuga, June 2003, Revised Papers, LNCS 2844, ISBN 3-540-20159-9.

Gould JD, Lewis C. 1983. Designing for usability. In *Human factors in computing systems*, Janda A (ed.). North Holland: Amsterdam; *Proceedings of the CHI'83 Conference*, Boston, MA.

Gould JD, Boies SJ, Ukelson J. 1997. How to design usable systems. In *Handbook of Human-Computer Interaction*, Helander M, Landauer TK, Prabhu P (eds). Elsevier Science BV.

Gulliksen J, Göransson B. 2001. Reengineering the systems development process for user centered design. In *Proceedings of Interact 2001*, Hirose M (ed.). IOS Press.

Gulliksen J, Göransson B, Lif M. 2001. A user-centered approach to object-oriented user interface design. In *Designing Interactive Systems: Object Modeling and User Interface Design*, Van Harmelen M (ed.). Addison-Wesley: Boston, MA, ISBN 0-201-65789-9.

Gulliksen J, Göransson B, Boivie I, Blomkvist S, Persson J, Cajander Å. 2003. Key Principles for User Centred Systems Design. Special section ''Designing IT for Healthy Work'', *Behaviour and Information Technology* **22**(6): 397–409.

Harris J, Henderson A. 1999. A better mythodology for system design. In *CHI 1999 Conference on Human Factors in*

Computing Systems Proceedings, Williams MG, Altom MW, Ehrlich K, Newman W (eds). ACM Press.

Hudson W. 2001. Toward unified models in user-centered and object-oriented design. In *Object Modeling and User Interface Design: Designing Interactive Systems*, Van Harmelen M (ed.). Addison-Wesley: Boston, MA, ISBN 0-201-65789-9.

ISO 9241-11 1998. *Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs). Part 11: Guidance on Usability*. International Organization for Standardization: Geneva, Switzerland.

ISO/IS 13407. 1999. *Human-Centred Design Processes for Interactive Systems*. International Organization for Standardization: Geneva, Switzerland.

ISO/TR 18529. 2000. *Ergonomics – Ergonomics of Human-System Interaction – Human-Centred Lifecycle Process Description*. First edition 2000-06-15, ref. number ISO/TR 18529:2000(E), International Organization for Standardization: Geneva, Switzerland.

Jacobson I, Booch G, Rumbaugh J. 1999. *The Unified Software Development Process*. Addison-Wesley Longman Inc: Reading, MA.

Jones JC. 1981. *Design Methods: Seeds of Human Futures*. 2nd edn. Wiley, London.

Kruchten P. 1998. *The Rational Unified Process – An Introduction*. Addison Wesley Longman Inc: Reading, MA.

Lewis CH. 1990. A research agenda for the nineties in human-computer interaction. *Human Computer Interaction* **5**: 125–143.

Lindegaard G. 2002. Deconstructing silos: The business value of usability in the 21st century. In *Usability Gaining a Competitive Edge*, Hammond J, Gross T, Wesson J (eds). Kluwer Academic Publishers: Boston, MA; *IFIP World Computer Congress 2002, Montreal, Canada.*

Mayhew DJ. 1999. *The Usability Engineering Lifecycle, A Practitioner's Handbook for User Interface Design*. Morgan Kaufmann Publishers: San Francisco, CA.

McCoy T. 2002. Usability: Who cares? In *Usability Gaining a Competetive Edge*, Hammond J, Gross T, Wesson J (eds). Kluwer Academic Publishers: 283–294; *IFIP World Computer Congress 2002*, Montreal, Canada.

Molich R. 2003. *Comparative Usability Evaluation – CUE*. http://www.dialogdesign.dk/cue.html, referenced August 14, 2003.

Nielsen J. 1993. *Usability Engineering*. Academic Press: San Diego, CA.

Nielsen J, Mack RL. 1994. *Usability Inspection Methods*. John Wiley & Sons.

Preece J, Rogers Y, Sharp H, Benyon D, Holland S, Carey T. 1994. *Human-Computer Interaction*. Addison-Wesley: Wokingham, UK.

Roberts D, Berry D, Isensee S, Mullal YJ. 1998. *Designing for the User with OVID: Bridging User Interface Design and Software Engineering*. Macmillan Technical Publishing: Indianapolis, IN.

Rosson MB, Carroll JM. 2002. *Usability Engineering: Scenario-Based Development of Human-Computer Interaction*. Morgan Kaufmann Publishers: San Francisco, CA.

Siegel D. 2003. The business case for user-centered design: increasing your power of persuasion. *Interactions*, Vol. X.3. ACM.

Siegel D, Dray S. 2003. Living on the edges: user-centered design and the dynamics of specialization in organizations. *Interactions*, Vol. X.5. ACM.

STANDISH GROUP. 1995. CHAOS Report. Available at ww.scs.carleton.ca/~beau/PM/Standish-Report.html, referenced August 14, 2003.

Sutcliffe A. 2002. *User-Centred Requirements Engineering: Theory and Practice*. Springer-Verlag: UK, ISBN 1-8523-3517-3.

Van der Veer G, Van Vliet H. 2003. A plea for a poor man's HCI component in software engineering and computer science curricula; after all: the human-computer interface is the system. *Computer Science Education* **13**(3): 207–226 (Special Issue on Human-Computer Interaction).

Wallace MD, Anderson TJ. 1993. Approaches to interface design. *Interacting with Computers* **5**(3): 259–278.

Wasserman AI. 1981. User software engineering and the design of interactive systems. *Proceedings of the 5th International Conference on Software Engineering*, San Diego, CA, 387–393.

Wasserman AI, Pircher PA, Shewmake DT, Kersten ML. 1986. Developing interactive information systems with the user software engineering methodology. *IEEE Transactions on Software Engineering* **SE-12**(2): 326–345.

Whiteside J, Bennett J, Holtzblatt K. 1988. Usability engineering: our experience and evolution. In *Handbook of Human-Computer Interaction*, Helander M (ed.). North Holland.