# Usability Throughout the Entire Software Development Lifecycle A Summary of the INTERACT 2001 Workshop

*Jan Gulliksen*
*Inger Boivie*

Department of Information Technology
Uppsala University
Box 337, SE-751 05 Uppsala, Sweden

**Technical report 2001-026**
November 2001
ISSN 1404-3203

# Usability Throughout the Entire Software Development Lifecycle
# A Summary of the INTERACT 2001 Workshop

## Introduction

The un-usability of systems, products and services is a tremendous problem for users and consumers all over the world, despite the efforts put in by researchers, usability practitioners and designers. Therefore, usability still needs to be the main focus of our activities. Research and development must focus more on developing processes, methods and tools that significantly turn IT development in another direction. In practice, usability aspects are usually regarded very late (if at all) in software development.

Software development does not stop with delivery, nor do usability issues. Most systems and products are modified and improved in a number of releases over a number of years. Web sites are continuously updated and modified. Most efforts at working with usability matters stop after the initial development process. What do we do after delivery?

Furthermore, commercial software development models, such as Rational Unified Process™ (RUP) and Dynamic Systems Development Method™ (DSDM) are becoming widely used in industry. Other software development models with different approaches are also starting to attract attention from industry, e.g. eXtreme Programming. These models are basically not user-centered and most of them provide limited support for usability activities. Thus, it is also important to find ways of integrating usability aspects into such development models.

To discuss these issues, the authors organised a one-day workshop at the INTERACT 2001 conference in Tokyo, Japan, on July 9, 2001. The workshop was an official workshop for the International Federation for Information Processing (IFIP) working group 13.2 on “Methodologies for User Centred Systems Design”. Nine position papers were accepted and the workshop gathered 10 participants. In addition, two representatives of IFIP Technical Committee 13 on Human Computer Interaction sat in on the workshop during parts of the day and contributed to the discussions. The position papers are included below.

The position papers were briefly presented at the start of the workshop, and the remaining time was spent discussing matters as described below.

## Previous Workshops

The first workshop on “User-centred design in practice – problems and possibilities” was held at the 1998 Participatory Design Conference (PDC’98) in Seattle, November 1998. The summary of this workshop was published in SIGCHI Bulletin [1] together with all the submitted contributions. This workshop highlighted the difficulties in adopting a fully user-centred design (UCD) approach in practice and the need to spend more efforts on making the UCD process work better. The workshop addressed among other things

- when and how to involve users in the design and development process
- practical experiences of prototyping and video recording in the analysis, design and evaluation processes
- organisational obstacles to user-centred design
- the role of the UCD facilitator in the development process
- communication problems that occur when people with varied skills and expertise communicate with one another

As a sequel to this workshop, another one on the topic “Making User-Centred Design Usable” was arranged at the INTERACT’99 conference in Edinburgh, Scotland, August 99. The summary of this workshop was also published in SIGCHI Bulletin [2]. This workshop focussed on the users of the user-centred design process, namely the software developers. The result of the workshop was a list of aspects that are crucial to the usability of the UCD process:

- communication
  the importance of meeting users and supporting a shared understanding.
- representations
  the need for understandable design representations and equivalent design representations, i.e. different representations that convey the same information about an object but in different forms and terms
- process
  the need for good qualified, experienced usability experts, as well as the necessity to cultivate IT in use. Each organisation must specify its own UCD process

- attitudes
  the importance of conveying UCD attitudes, not just tools and methods. UCD must be escalated to management level, by means of, for instance, business cases. One way is to create a demand for usability guarantees on the consumer/user side.

Based on the results of these two workshops, we felt a need to arrange a follow-up workshop addressing usability throughout the entire software development .

## Position Papers – Submissions and Presentations

The position papers and the presentations at the workshop are summarised briefly below. The full versions of the papers are attached in this document, and also available on the workshop website. (www.hci.uu.se/~jg/UCD2001/)

- *Unifying User-Centered and Use-Case Driven Requirements Engineering Lifecycle* – Antunes H., Seffah A., Radhakrishnan T. and Pestina S.
  The paper discusses the problem of where to integrate usability activities in a software engineering process and the need for improving and mediating software-usability engineering communication. They suggest re-designing the software engineering, represented by Rational Unified Process (RUP), to include users and usability expertise. They compare the forms used in RESPECT [3] to describe the context of use and tasks with use cases. The authors argue that the same artifact should be represented in several notations to serve the different needs in the software and usability engineering processes.
  During the presentation Seffah argued that one common view on the relation between software engineering and usability is that the developers build the system, then the usability experts make it usable. Nothing could be more wrong! Such an approach does not take into account the relation between the external and the internal parts of software. Previous attempts to integrate usability activities and software development have been done in particular contexts. A formal, general, framework for integrating usability into basically any software development process is needed.

- *An Evaluation Framework for Assessing the Capability of Human-Computer Interaction Methods In Support of the Development of Interactive Systems* – Daabaj Y.
  Daabaj suggests a framework for evaluating the applicability of task analysis methods in software development, particularly in the requirements capture phase. The task analysis methods are evaluated against criteria on, for instance, the usability of the output, the scope of analysis, representation format and requirements mapping. Unfortunately the author could not participate in the workshop to elaborate on his paper.

- *Incorporating Usability into an Object Oriented Development Process* – Ferré X.
  This paper discusses usability engineering and use cases in object-oriented software development. Ferré argues that use cases may bridge the gap between software engineering and usability engineering but they need a supplementary user-centred focus. The author suggests a joint usability/software engineering development cycle based on Larman's object-oriented approach.
  Ferré argued that there is too much focus on technology and low-level design in software engineering, resulting in too many design decisions being made during analysis. It is difficult for developers not to think in design terms when modelling, since the models will eventually be turned into designs. Use cases may be the solution to this problem. Use cases are, however, not used properly from a usability point of view. They are often seen as an early version of design artifacts, taken from the user to the designer/technology world. The focus should move from requirements to design, where the external design must come before the internal design. The shift to a design focus calls for less formality in the software development models. At the same time, usability activities move towards a higher degree of formality.

- *Modelling the Usability Testing Process with the Perspective of Developing a Computer Aided Usability Engineering (CAUE) System* – Gellner M. and Forbrig P.
  This paper discusses the need to create support for usability evaluation activities in software development projects. The authors compare the evaluation process to the software engineering process and suggest a framework consisting of eight phases to describe the evaluation process. The authors outline a computer support for usability testing (CAUE). This tool should support all the eight phases

of the evaluation process and be aimed primarily at organisations without usability expertise. The below figure illustrates the evaluation process.

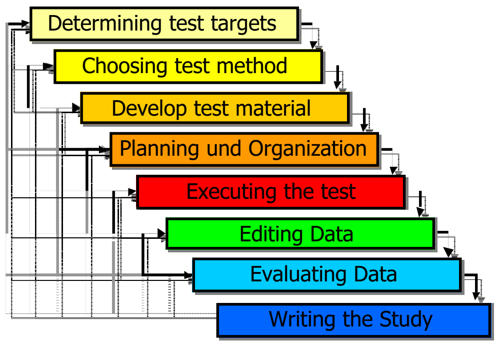


*Fig1: The eight phases in the usability evaluation process*

- *A Usability Designer at Work* – Göransson B.
  Göransson describes the usability designer role, which merges ideas from usability engineering as described by, for instance, Nielsen and the interaction design approach suggested by Cooper. The usability designer participates continuously throughout the entire development project. He/she has the responsibility for the user-centred approach and all usability-related activities in the project, including taking an active part in the design process.
  Göransson particularly wants to address the problems with the "usability-at-the-end" view and the fact that many commercial software development models do not honour the importance of usability. Usability is taken for granted. He also argues that user-centred design, a prerequisite for usability, is about attitudes and process, with an emphasis on attitudes. User-centred design ought to be the standard operating procedure for software development, and the usability designer provides a way of achieving that.

- *Usability as a Tool for Competence Development* – Holmlid S.
  This paper discusses a model for use quality to be used in design and the setting up of a learning environment. The concept learner-centred design is introduced and the role of the learner facilitator is discussed. Holmlid describes a project which has been performed in cooperation with a major Swedish bank. Unfortunately the author could not participate in the workshop to elaborate on his paper.

- *Learning from traditional architects* – Johnston L.S
  Software development is an engineering discipline, and may as such, be compared to other engineering disciplines. Johnston discusses the roles of the architecture in building projects throughout history. When architects were absent in the building process during industrialisation a low standard of building was the result. The engineers, assuming the role of the architect, put functionality before form and people. Presently, the role of the architect has changed into being more like a user representative and context of use analyst. In addition to the architect being the users' advocate, each project needs a good project manager to see to it that the process is doing what it should do. In addition, "up front" quality goals have to be specified
  The author also brings up usability patterns. She argues that they may help capturing good practice, reducing the need for iterations.
  During the presentation, Johnston also pointed out that there is a large difference between small-scale development and development on a large scale. She also brought up the question about what needs to be shared between HCI people and software engineers.

  *Evidence-Based Usability Engineering: Seven Thesis on the Integration, Establishment and Continuous Improvement of Human-Centred Design Methods in Software Development Processes* – Metzker E.
  Evidence-based usability engineering is based on three fields; usability engineering, software process improvement and knowledge management. Metzker outlines a tool for compiling know-how and expertise regarding HCI methods and tools. The aim is to increase the effectiveness of usability

activities and to support a flexible strategy for integrating usability engineering in software development processes. The tool is based on the idea that the efficacy of HCI activities depends on the development context. Thus, HCI methods and processes cannot be introduced into an organisation as a fixed workflow model. They have to be adapted to the particular context. The suggested tool would be based on available evidence of the usefulness and applicability of a particular method or technique. It would provide support in choosing and adapting methods and techniques to the situation at hand. See the illustration of the evidence-based usability engineering process below.

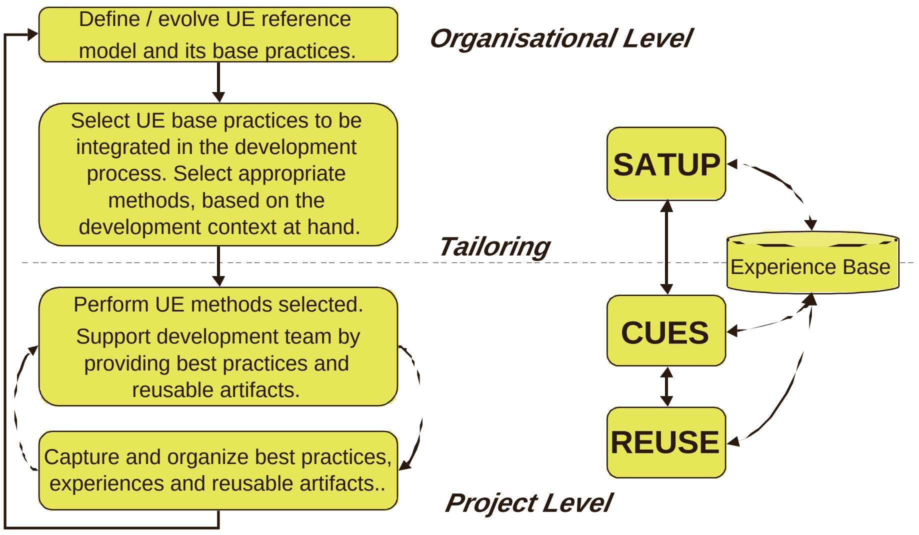

Define / evolve UE reference model and its base practices.
Organisational Level
Select UE base practices to be integrated in the development process. Select appropriate methods, based on the development context at hand.
SATUP
Tailoring
Experience Base
Perform UE methods selected. Support development team by providing best practices and reusable artifacts.
CUES
Capture and organize best practices, experiences and reusable artifacts..
REUSE
Project Level

*Fig 2: Evidence-based usability engineering process*

- *User Intelligence Will Make Mobile Solutions Fly* – Olsson A. and Svantesson S.
  The authors describe an approach to capturing and compiling information about the user and the usage of mobile applications. User intelligence consists of frame-finding (who are the users, what do they do, where, when, how and with what) and gaining insight (goals, problems, desires and values). Common data collection methods are interviews, observations, videos, etc. The approach also includes prioritising features, creating scenarios, prototyping and qualitative usability evaluations.
  In the workshop, the authors emphasised the importance of usefulness in products. A product must add value to peoples' lives. People will not use products if they do not feel the need of the services. They also mentioned the necessity of communicating the impact of user experience to the project managers. What is it like being a user? What is user experience? The authors work at a Swedish consultancy and have introduced and established the user intelligence approach in the organisation which has taken them 3-4 years to achieve. Currently, an information designer typically works throughout the development process, controlling it. The use cases are written by the information designer and software engineers to make sure that both views are represented.

## Usability Throughout the Entire Software Development Lifecycle

What do we mean by usability throughout the entire software development? What expertise, aspects, issues, activities, etc, should be included in this approach? If we were to write a textbook on usability in the entire development lifecycle, what topics ought to be included? The participants were invited to list the topics that they considered the most important for such a book. The contributions were then categorised as described below.

The ensuing discussion primarily concerned the target group and main aim of such a book. Who should be the intended reader, software engineers or HCI people? Is the aim to introduce usability and increase the usability awareness among software developers, or is it to facilitate the introduction and establishing of usability in an organisation? The discussion about the target readers and the main aim of such a book highlighted the problems and tensions within the field. Is usability a concern of the HCI expert participating in the projects, or should it be the concern and responsibility of the software engineers? Some of the participants argued that presently, there are no textbooks on usability that cater to the needs and background knowledge of software engineers. Is it so, perhaps, that few within the HCI community have enough knowledge about software engineering to address the issues and problems with usability that pertain to engineering? In the HCI community, usability is the major aspect within software engineering, requiring special attention, expertise and methods based on, for instance, psychology and ethnography. For the software engineer, usability is one aspect out of many that must be taken into account, preferably by means of an engineering approach. One interesting matter was that the representatives of the software engineering community in the workshop favoured an engineering approach to usability – the papers submitted by them mainly related to the usability engineering approach suggested by Mayhew [4], etc.

The topics suggested for a book on usability in the entire lifecycle were

- *Definitions*
  A textbook on usability in software development must provide good, applicable, agreed-on definitions and descriptions of a number of basic concepts. What is usability and how does it relate to and affect software development? What is user-centred design? What is usability engineering? A roadmap of how user-centred design, usability engineering and the HCI field relate to one another was suggested. Furthermore, some of the fields contributing to HCI must be covered, for instance, ethnography and cognitive science.
- *Process*
  A section on the development process should discuss different models for integrating usability in the software development lifecycle. One particular matter of interest is whether usability activities should be a separate process running in parallel with or be an integral part of the software engineering process. Most participants were in favour of a tight integration. Typical usability activities, tools and methods should be described briefly as well as concepts such as iterative development and incremental development. Another important aspect to cover is how to introduce and establish usability activities in mainstream software engineering processes. How formal must a process be, how to best manage it, and how do we maintain the usefulness and usability focus throughout the development? Time aspects are important and the book should cover matters, such as, when to start the actual design and how much time to spend on capturing requirements in relation to other activities.
- *Roles and responsibilities*
  Some of the roles suggested were project manager, software engineer and usability expert or usability engineer. What are the responsibilities and activities of each role? Does usability require an overall role, a usability champion? Does usability require expertise on the client side as well as on the development team?
- *Users*
  The users are so important that they deserve a whole section of their own. It is essential that the software development team identify the users and other stakeholders, and understand who their users are. Different methods for investigating users' needs, situation, behaviour, interest and motives should be described, including methods for evaluating the user experience. The matter of involving users, if, when and how must be covered. Problems that may arise from large user groups and broad varieties of users should be addressed.
- *Requirements*
  How do you best capture and understand the needs of the users? How do you ensure usability and

quality? Is it possible to order a usable system? This section should cover topics, such as, usability requirements/goals and functional requirements, requirements engineering with a "usability flavour" and requirements elicitation methods.

- *Tasks and use cases*
  Understanding the work to be supported by the system is an essential part of designing for usability. The book should cover different methods for analysing and describing tasks. The HCI field provides a number of models for analysing and describing human activities, for instance, hierarchical task analysis, activity theory and situated action. The applicability of such methods in software design is, however, much questioned. Other approaches have been proposed by the software engineering community, in particular use case modelling applied in object-oriented development methodology.
- *Design*
  Design has been somewhat of a blind spot within the HCI community, where a majority of the proposed usability methods and techniques focus on analysis and evaluations. The importance of the design phase has been increasingly acknowledged over the last few years, however. A design section should cover design methods, for instance, contextual design, scenarios, prototyping techniques and criteria-based design. This section should also discuss conceptualisation and how to design the information structure/architecture. Design patterns - what they are, how they may be used, advantages and disadvantages, etc - should also be included.
- *Evaluations*
  Usability evaluations ought to be an important part of the usability efforts in any software development project. This section should describe different methods for usability testing and inspections, their main characteristics and usage, the advantages and disadvantages of each method, etc. How to plan and set up an evaluation, report the findings and feed them back into the design are important matters.
- *Tools*
  This section should cover tools for supporting the usability activities in a software development project, in particular such tools that address the gap between the usability expertise and the CASE tools used by the software engineers.
- *Project aspects*
  The software development process must be adapted to the undertaking at hand. Criteria that determine the development process include the size of the project – small-scale versus large-scale projects – the domain, life expectancy of the software, quality requirements, etc.
- *Organisational aspects*
  Depending on the intended reader group the book should contain a section on how to introduce and organise the usability work in an organisation. This includes specifying a framework for selecting, defining and evaluating the integration of usability and user-centred design in software engineering. Practical applicability, industry practices and cost-justification should be addressed, as well as cultural aspects.
- *Communication*
  Communication is a recurring theme in our workshops. Good communication between users, software developers, project managers, client representatives, etc, is a prerequisite for success in software development. One important aspect is the creation of a shared understanding of the problem to be solved.

It may seem an impossible endeavour to write such a comprehensive book, but the reader must bear in mind that the list above is the sum total of all the contributions from all the participants. Should any one of the workshop participants undertake to write a textbook, the contents would certainly be adapted to the special interest of the intended reader group or groups, as well as the particular hobbyhorses of the author.

## Separate Track or Seamless Integration

Many projects start without the intended users – they start with the functional requirements and client requirements expressing the business goals rather than the user requirements. The ideal would be to start with a group containing user representatives, usability experts and software engineers, even before the requirements capture phase starts. Requirements can then be captured by means of user research.

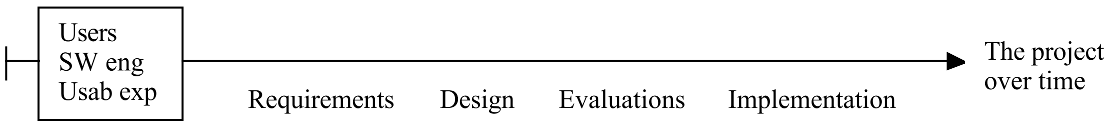


*Fig 3: Development cycle*

Users and designers need to get to know each other. Users must be involved early and continuously and particularly in the design phase. Some of the participants argued that it is easier to involve users in interface design activities than in use case modelling. Does this mean that the interface should be designed before the use case modelling starts? The matter will be further discussed in the section on tasks and use cases below.

Do we need a separate usability process or should it be seamlessly integrated into the software development process? The group did not reach agreement on the matter, but everyone at least agreed that it is a key problem. Why do we need a usability process at all? Processes are rarely used to guide the day-to-day work and most people only care about their own parts of the process. Processes are, however, useful for reasoning about one's work and describe it to others. They are particularly important for persuading management to introduce usability activities in software development. Having a separate usability process may help in making the software development process more suitable for a usability focus.

## Patterns

Design patterns or usability patterns have raised the hopes amongst software designers and the HCI community of late. With patterns, good design solutions may be re-used, and bad ones hopefully eliminated. Design patterns could be seen as user needs resolved into working solutions, representing best design practices.

Patterns can be used to facilitate communication between different groups, but need not be understood by users. One concern is finding a design pattern to fit a particualar design problem – naming design patterns suitably is therefore important. The names of the patterns should help jog the imagination of the designer.

There are a number of product-based design patterns and pattern languages. Pattern languages are groups of patterns that capture a philosophy of solutions. The patterns in a language share the same design philosophy and come from the same design rationale. A pattern language can be domain specific or system type specific. Pattern languages were further discussed at the INTERACT 2001 conference by Mahemoff and Johnston, in the paper Usability Pattern Languages: the "Language" Aspect.

Design patterns or usability patterns are typically product-oriented – but methods and techniques could be seen as process-oriented patterns. The USEPACKs (Usability Engineering Experience Package) discussed by Metzker in his position paper, *Evidence-Based Usability Engineering: Seven Thesis on the Integration, Establishment and Continuous Improvement of Human-Centred Design Methods in Software Development Processes,* could be seen as a kind of patterns. A USEPACK is used to capture best practices in applying various HCI methods and techniques. It is a semi-formal notation and describes the HCI activity or method with an increasing level of complexity and detail. The USEPACK describes the activity/method, the development context in which it is suitable and provides a set of artifacts such as checklists and templates. The idea is to provide guidance and inspiration in such a way that USEPACKs can be mapped into the software engineering process also by less experienced usability workers.

## User-centred Design versus Usability Engineering

Are usability engineering and user-centred design the same? Or is it so that engineers equate usability engineering with user-centred design, just because of the engineering suffix? Do they need an engineering suffix in order to accept a user-centred design approach? The computer scientists participating in the workshop said that to many software engineers, user-centred design is just another word for usability engineering. Software engineering is far from being a mature discipline, it certainly needs further research and development as regards user- orientation.

User-centred design is a philosophy opposed to the system-driven development philosophy that is the traditional way of seeing and doing things in software development. User-centred, or human-centred design is the way we want to move in software development, but it has not become established practice. Hopefully, it will become the established, traditional way of doing things. However, to dislodge the system-oriented approach takes an enormous amount of effort or a miracle. It may be on its way with new technologies and decentralised software development.

It is important though, to use the terminology unequivocally. It is perhaps a bit unfortunate that usability engineering has become the way of thinking about user-centred design in the software engineering community. Usability engineering focuses on requirements and evaluations preserving, perhaps, a technical, engineering-oriented attitude to software development. User-centred design, on the other hand, addresses *designing* with the users. Seffah presented a slide clarifying the differences between human-centred and technology-driven development, see the illustration below.

| **Traditional software development philosophy** | versus | **Human-centered development philosophy** |
|---|---|---|
| Technology/developer-driven | → | User-driven |
| Component focus | → | Solution focus |
| Individual contribution | → | Multidisciplinary teamwork including users, customers, human factors experts |
| Focus on internal architecture | → | Focus on external attributes (look and feel, interaction) |
| Quality measured by product defects and performance (system quality) | → | Quality defined by user satisfaction and performance (quality in use) |
| Implementation prior to human validation | → | Implementation of user-validated solution only |
| Solutions are directed by functional requirements | → | Understanding the context of use (user, task, work environment) |

*Fig 4:Traditional software development versus human-centred developmen*

## Use Cases Versus Task Analysis

Use cases provided the hottest topic of the day, causing a rather heated discussion. Despite the fact that the group that discussed use cases and tasks during the breakout session could not agree on what a use case is. Use cases describe what the *system* does and define its limits including task allocation – but how? The four participants represented four different views – from a use case being just a name of a task to a use case being a detailed specification of the task including interaction details, such as button clicks. The level of description should be determined by usability concerns. You need a good task model in order to create a usable system.

Who should specify the use cases? Dedicated use case specifiers, software engineers or the usability expert? Should users be involved? The group could not agree on whether or not to involve users. Some participants argued that use case modelling was introduced into software development *in order to* facilitate early and continuous user involvement. Use cases are meant to replace functional/technical approaches in requirements elicitation.

Use cases supposedly provide the means for getting the user needs into the projects in that they capture the functional requirements in terms of what the user wants to do with the system. It was argued that use cases are a collection of functionality and that such a collection must be the starting point in the project. You need the functional requirements before starting the user interface design. Use cases capture what the software engineers need to know about the context and the user needs. They are the starting point for the software engineers.

Based on their own experiences, other participants argued that users and clients (and sometimes software developers!) often do not understand how to write use cases. They are difficult to explain and to apply. Users generally find it easier to participate in interface design than in use case modelling. As somebody pointed out:

"Writing use cases with clients, what you get is an approved use case model. But when the design comes out the client does not want that."

The use case sceptics argued that user interface design, instead of use case modelling, should be the starting point of the development project. The use case advocates, however, objected to this approach. Use case modelling, they argued, must precede user interface design. This is clearly a point on which usability people and software engineers differ. Software engineers start with functionality and usability people with the user interface (interaction). One participant pointed out that interface prototypes and use cases should be used together and that a conceptual model of the interface should be developed before starting on the use cases. There are also alternative use case approaches, in particular essential use cases [5] that are supposed to better support user interface design in that they do not contain interaction details.

Do we need use cases at all? They seem to present quite significant problems. How do we move on from use cases to integrate all the aspects that are necessary to usability? Use case models are used to facilitate communication between the different parties in software development, but they are not good enough. Obviously, we need something else to bring users, usability and software engineers together. Currently available alternatives are storyboards, scenarios, design mockups, prototypes, etc. There are also a number of methods for working with the design phase, such as, participatory design workshops, parallel design workshops and preference-based design.

## Conclusions

Usability being a natural part of software development is still far away, though some organisations are slowly closing in on the ideal situation. The workshop contained many fruitful discussions and some provocative thoughts were expressed. Some of the conclusions that we have arrived at from the workshop discussions are described below.

- Get software engineers and HCI people together
  It is imperative to further deepen the cross-disciplinary work between the HCI community and the software engineering community. Views on user involvement, user-centred design, work context analysis, etc tend to differ. The discussions further pointed to the fact that much of the literature about usability is produced within the HCI community and unsuccessful in addressing the concerns of the software engineers in relation to usability. There are basically no textbooks on HCI and usability that can be used in the software engineering education!
- Use cases are not the answer to all problems in software design
  Use cases versus task analysis and use cases versus interface design are obviously topics that need elaboration. Moreover, they particularly call for software engineers and usability people getting together to discuss the problems. Or we will end up with two camps, each spouting arguments for their own convictions and the users getting nothing out of it. How do we best capture the routines, procedures, events, actions – in short, the details that make up a work context? Are the use cases the answer, and in such case, how should they be complemented to better support user requirements capture, usability efforts and interaction design? Should use cases be replaced by something better? What?
- Future work
  Given the fact that most software engineers are introduced to the field of HCI through books on

usability engineering (e.g. Nielsen, 1992, Mayhew, 1999, Faulkner, 2000) or from more software based methodologies (e.g. Constantine & Lockwood, 1999) there is a need for material that communicates the knowledge and experience established within the field of HCI in a better and more direct way. The discussion after the workshop has driven us to propose an edited book integrating HCI and software engineering (proposal to come). Hopefully such a book could fill the need to educate software engineers on usability in a way that they can collaborate with HCI experts. Furthermore we will pursue the discussion on upcoming working conferences and workshops organised by IFIP WG 13.2 on the topic. For more information please watch the upcoming website for IFIP WG 13.2. (www.hci.uu.se/~jg/IFIP13.2/)

## References

1. Gulliksen J., Lantz A. and Boivie I. (1999). *User Centered Design – Problems and Possibilities*. SIGCHI Bulletin, Vol. 31, No. 2, April 1999, pp. 25-35. Summary of the PDC '98 workshop on User Orientation in Practice – Problems and Possibilities. Also available with all accepted contributions as technical report TRITA-NA-D9813, CID-40. (www.nada.kth.se/cid/pdf/cid_40.pdf)
2. Gulliksen J., Lantz A. and Boivie I. (2001). How to make User Centred Design Usable. SIGCHI Bulletin, Vol 33, No. 3, July 2001. Summary of the INTERACT '00-workshop on How to make user centred design usable. Also available with all accepted contributions as technical report TRITA-NA-D0006, CID-72. (http://cid.nada.kth.se/pdf/cid_72.pdf)
3. Maguire M.C. (1998). *RESPECT User Centered Requirements Handbook*. EC Telematics Applications Program, Project TE 2010 RESPECT, Wp5 Deliverable D5.3, 1998
4. Mayhew D.J. (1999), *The usability engineering lifecycle, a practitioner's handbook for user interface design.* Morgan Kaufmann Publishers Inc., San Fransisco, CA.
5. Constantine, L.L., Lockwood, L.A.D., (1999) *Software for Use – A Practical Guide to the Models and Methods of Usage-Centered Design*, Addison Wesley Longman, Inc. Reading, Massachusetts.
6. Nielsen, J. (1993). *Usability Engineering.* Academic Press, Inc., San Diego.
7. Faulkner, X. (2000) *Usability Engineering*, Palgrave, New York, N.Y., ISBN 0–333–77321–7.
8. Constantine, L. L. & Lockwood L. A. D., (1999), *Software for Use: A Practical Guide to the Models and Methods of Usage-Centred Design*, Boston, Addison-Wesley.

## Position Papers

Included are the position papers submitted to and accepted at the workshop.
They are

- Unifying User-Centered and Use-Case Driven Requirements Engineering Lifecycle – Antunes H., Seffah A., Radhakrishnan T. and Pestina S.
- An Evaluation Framework for Assessing the Capability of Human-Computer Interaction Methods In Support of the Development of Interactive Systems – Daabaj Y.
- Incorporating Usability into an Object Oriented Development Process – Ferré X.
- Modelling the Usability Testing Process with the Perspective of Developing a Computer Aided Usability Engineering (CAUE) System – Gellner M. and Forbrig P.
- A Usability Designer at Work – Göransson B.
- Usability as a Tool for Competence Development – Holmlid S.
- Learning from traditional architects – Johnston L.S
- Evidence-Based Usability Engineering: Seven Thesis on the Integration, Establishment and Continuous Improvement of Human-Centred Design Methods in Software Development Processes – Metzker E.
- User Intelligence Will Make Mobile Solutions Fly – Olsson A. and Svantesson S.

# *Unifying User-Centered and Use-Case Driven Requirements Engineering Lifecycle*

**H. Antunes, A. Seffah, T. Radhakrishnan, S. Pestina**

*Department of Computer Science, Concordia University,*
*1455 de Maisonneuve W., Montreal PQ H3G 1M8, Canada*

Tel: *1 (514) 848 3024*
Fax: *1 (514) 848 2830*
Email: *{antunes, seffah, krishnan, pestina}@cs.concordia.ca*
URL: *http://www.cs.concordia.ca*

**In the last five years, many software development teams have tried to integrate the user-centered design techniques into their software engineering lifecycles, in particular in the use case driven software engineering lifecycle. However, because of lack of understanding and communication between two diverse teams and cultures, they often run into problems. One problem arises from the fact that the software engineering community has their own techniques and tools for managing the whole development lifecycle including usability issues, and it is not clear where exactly in this usability engineering techniques should be placed and integrated with existing software engineering methods to maximize benefits gained from both. This paper identifies the principles of a cost-effective communication line between human factors/usability specialists and software development teams. It also describes a tool that can help to understand, define and improve this communication line while facilitating the integration of usability in the software development lifecycle. As a case study, we will consider two popular requirements engineering processes: user-centered requirements process as defined in ISO-13407 and implemented in RESPECT and the use case driven requirements process as defined and implemented in the Rational Unified Process.**

**Keywords:** usability engineering, software development lifecycle, use cases user-centered design, human-to-human communication.

## 1 Introduction

For small-size projects, software development teams can mostly avoid the direct involvement of usability experts, due in particular to the availability of design guidelines and usability patterns, heuristics for evaluation or tasks flowcharts to supplement the functional requirements analysis. However, for large-scale projects it is necessary, almost impossible, not to involve explicitly usability specialists, at least during the requirements analysis and usability testing steps. Culled from our day-to-day experience, four different ways, for involving usability expertise in the software development teams, are possible: (1) resort to third part companies specialized in usability engineering, (2) involve a consultant expert in usability, (3) form/create a usability team, and finally (4) provide training to some members of the development team that can act as the champions of the usability.

However, whatever the approach chosen for involving usability engineers in the software development lifecycle, the difficulties of communication between the software development team and the usability specialists could seriously compromise the integration of the usability expertise in software development lifecycle. Among the difficulties of communication, one can mention the educational gap, the use of different notations, languages and tools, as well as the perception of the role and importance of the design artifacts. For example, in spite of the similarities existing between use cases and task analysis (Artim et al., 1998; Forbrig, 1999; Seffah et al., 1999) and the advantages by their complementarity uses, the software and usability engineers often try to substitute one by other.

The ultimate objective of our research is to build a framework, while contrasting and comparing the software and usability engineering lifecycles, for improving and mediating the communication between the software development teams and usability engineers. This framework is governed by the questions below we are addressing:

- How can the software engineering lifecycle be re-designed so that end users and usability engineers can act as active participant throughout the whole lifecycle?
- Which artifacts collected and generated in the usability engineering lifecycle are relevant and what are their added values and relationships with software engineering artifacts?
- What are the usability techniques and activities for gathering and specifying these relevant artifacts?
- How can these artifacts, techniques and activities be presented to software engineers (notations), as well as integrated (tool support) in the software development lifecycle in general?

## 2 Background and Related Work

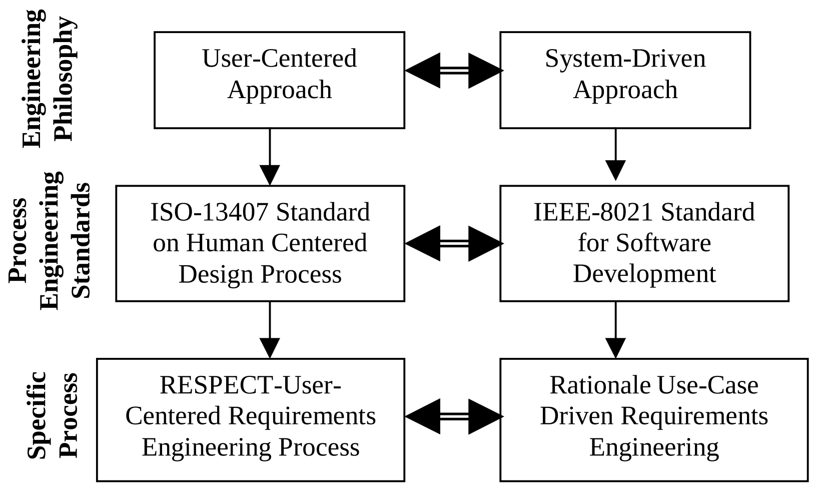
The following are only some of the many investigations that, over the last few years, have tried to answer such questions.

Artim (1998) emphasizes the role of task analysis by providing a user-centric view of a suite of applications, and then emphasizes use cases by providing each application with a method of exploring user-system interaction and describing system behavior. Jarke (1999) points out that scenarios are used in software engineering as intermediate design artifacts in an expanded goal-driven change process. They provide a task-oriented design decomposition that can be used from many perspectives, including usability trade-off, iterative development and manageable software design object models. Ralyte (1999) in the CREWS project develops a framework for integrating different kinds of scenarios into requirement engineering methods. Constantine (1999) suggests that use case specifiers first prepare lightweight use case model descriptions (*essential use cases*) that do not contain any implicit user interface decisions. Later on, the user interface designer can use these essential use cases as input to create the user interface without being bound by any implicit decisions. Nunes (1999) proposes to annotate use cases using non-functional requirements at the level of abstraction at which they should to be considered. Rosson (1999) proposes combining the development of tasks and object-oriented models, which are viewed as a refinement of rapid prototyping and an extension of scenario-based analysis. Krutchen (1999) introduces the concept of *use case storyboard* as a logical and conceptual description of how a use case is provided by the user interface, including the interaction required between the actor(s) and the system.

## 3 A brief Description of the Processes Investigated in Our Case Study

As starting point of our investigations and a research case study, we are considering the following two requirements processes (Figure 1):

- The use case driven requirements workflow as defined in the Unified software engineering Process (UP) proposed by Rational Software Inc (Booch, 1999).
- The RESPECT framework (REquirements SPECification in Tematics), which is concerned with the capture and specification of end-user requirements (Maguire, 1998).



Engineering Philosophy
Process Engineering Standards
Specific Process
User-Centered Approach
System-Driven Approach
ISO-13407 Standard on Human Centered Design Process
IEEE-8021 Standard for Software Development
RESPECT-User-Centered Requirements Engineering Process
Rationale Use-Case Driven Requirements Engineering

Figure 1. A View of our Research Case Study and Framework.

### *3.1 Capturing User Requirements as Use Cases in the Unified Process*

The goal of the requirements process, as defined in the unified process (UP), is to describe what the system should in terms of functionalities do in terms of functionalities, and allow the developers and the customer to agree on this description. Use cases are the most important requirements artifact. It is used by (1) the customer to validate that the system will be what is expected in terns of functionalities, and (2) by the developers to achieve a better understanding of the requirements and a starting point for technical design. A Use case storyboarding, which is a logical and conceptual description of how use cases are provided by the user interface, includes the required interaction between the user(s) and the system. Storyboards represent a high-level understanding of the user interface, and are much faster to develop than the user interface itself. The use case storyboards can thus be used to create and evaluate several versions of the user interface before it is prototyped designed and implemented (Krutchen, 1999).

One of the weaknesses of use case driven requirements workflow is that the use cases attempt to describe representative ways in which the user will interact with the software but is not comprehensive. Another weakness of this process is that the main people involved in this process are stakeholders and technical persons including use case specifier and user interface developer. End-User is not directly involved. Use case specifier details the specification for a part of the system's functionalities by describing the requirements aspect of one or several use cases.

### *3.2 User Requirements Engineering in RESPECT Framework*

RESPECT is a user-centered requirements engineering framework developed by European Usability Support Centres. The RESPECT process is a concrete implementation of the iterative user-centered design process for interactive software suggested by the ISO-13407 Standard (ISO, 1999). The RESPECT process starts from the point where project is summarized from the end user point of view. By the end of the process, it produced different text-based forms that detail the user interface, user support and help, the physical and organizational context, equipment and hardware constraints, usability goals that must be achieved, as well as the system installation procedure.

Although RESPECT is a highly detailed process for capturing and validating context of use and usability requirements with the active involvement of end-users and stakeholders, the text-based forms produced are not easily understandable by software development teams. They are also a source of ambiguity and inconsistency, especially when they are compared to the use cases.

## 4 Principles for Improving the Human-to-Human Communication

Our first step for improving and mediating software-to-usability communication involved identifying complementarities between the use-case requirements and RESPECT processes. The four principles outlined below summarize these complementarities.

Firstly, RESPECT captures a complete description of the context of use including user characteristics, task analysis, as well as the physical, technical and organizational environments in which the system will be used. Although in theory use cases have the potential to gather the non-functional requirements that are a simplified description of the context of use, in practice, use cases have been used for gathering the system functionalities and features including technical capabilities and constraints. Therefore:

> **Principle 1:** Context of use and functional requirements should be considered as two views of the requirements picture. The software view on this picture is a set of artifacts describing the functionalities and the technical requirements of the system. The usability view is a set of artifacts describing the context of use and the usability goals/factors in which the functionalities will be used.

To a certain extent, this principle means that both the software and the usability views are important. Table 1 indicates the software and usability views for each of the processes that we considered in our case study. Such classification of the artifact can facilitate the identification of potential relationships between artifacts.

|  | RESPECT | UP Requirements Workflow |
|---|---|---|
| Software View | General system characteristics<br>System functions and features<br>User Interface | Use Case Diagram<br>Requirements attributes<br>Boundary class<br>Use case storyboard<br>User interface prototype |
| Usability View | Organizational structure<br>Task scenario and interaction steps<br>Technical environment<br>User support<br>Physical environment<br>Social and Organizational environment | Stakeholder and Users needs<br>Additional Requirements |
| Other artifacts that cannot be classified. | Standards and style guides to apply<br>Test plan<br>Implementation plan | Vision document<br>Glossary |

Table 1. Relationship between RESPECT and UP Requirements Artifacts

Secondly, in RESPECT, the context of use is described using a non-formal notation which is easy to understand by end-users and stakeholders. However, these forms are a cause for inconsistency and ambiguity when used by software developers. The artifacts that are produced and the semi-formal notation used in use case approach are more understandable by software developers. Use cases as a notation can also support, in a certain extent, automatic generation of code (Krutchen, 1999, Booch, 1999).

**Principle 2:** As Artim's (1998) discussed about "one model, but many views and notations." We strongly share his belief that different notations for the same concept may foster communication between persons. This means that we can use different notations to describe the artifacts related to the functional and context of use including text-based forms and use cases. However, this requires maintaining the correspondence between multiple views at an abstract level using a high level notation.

Thirdly, in RESPECT as in other similar approaches, usability specialists use the *context of use* as an important input for usability testing. Software developers use the functional requirement artifacts as a starting point for technical design and implementation.

**Principle 3:** A common step to the two processes should include activities for reviewing and validating the integrity and consistency of all requirements artifacts from both the usability and software views. After validation, we should generate a usability testing and implementation portfolios.

For example, the usability-testing portfolio should include the entire usability requirement artifacts that will be used during usability testing. The implementation plan should include the artifacts that required for implementing the system.

Fourthly, it is important for usability-to-software engineering collaboration and for consistency and coherence of requirement artifacts to gain a high-level understanding of the system and this from the beginning. Therefore:

**Principle 4:** The requirements should start when a representative set of users and/or stakeholders are invited to summarize the system from the future user's perspective. They are mainly asked to answer different questions that we organized in a system summary form. Users and stakeholders, the main contributors during this step, are invited to give brief answers to these questions. All completed forms are then analyzed and compiled in a unique system summary form by usability engineers. This compiled form is approved by software developers, stakeholders and users. It is used as a roadmap during the requirement process and represents a general consensus on the system.

| Questions | Assumptions |
|---|---|
| What is the purpose of the system? | ISO 9000-based quality system over an Intranet |
| Why is this system necessary? | Supporting the development of the company outside the country (new clients, remote offices.) |
| Who will use the system? | Employees and some of the company's clients |
| What will the users accomplish with the system? | Access to quality procedures and associated forms<br>Learn the quality system and the ISO 9000 standard |
| Where will the system be used? | Standalone workstations and personal digital assistants |
| How will users learn to use the system? | Introductory course and online assistant |
| How will the system be installed? | By a Webmaster for the server version, and by employees on their PDA (download from the server) |
| How will the system be maintained? | By a Webmaster and a quality control manager |

Table 4. An Example of the System Summary Form.

Table 4 is an example of the system summary form that we developed. User-centered requirements frameworks such RESPECT and use case-driven approach supporters (Constantine, 1999) suggested similar questions.

## 5 A Framework for User-Centered and Use-Case Driven Requirements Engineering

Based on these principles, we iteratively defined, used and validated a framework for improving software-to-usability engineering communication (Figure 2). This framework clarifies how usability expert activities can be incorporated in the software development lifecycles. It also clarifies the relationships between activities done by software engineers and usability experts.

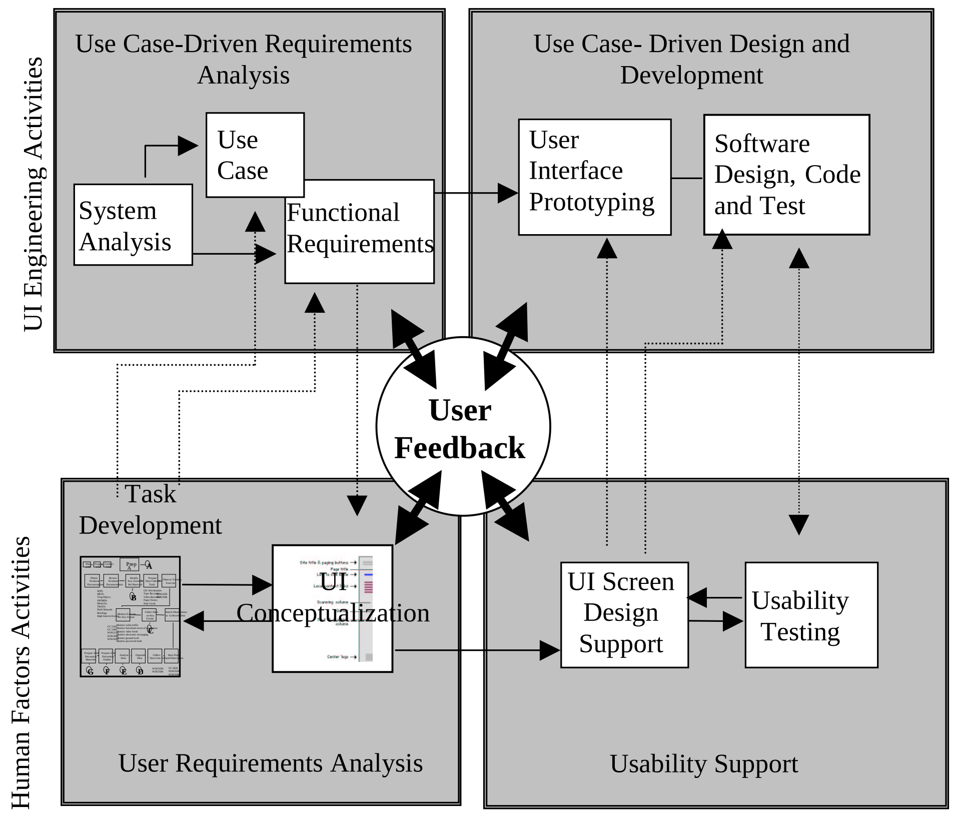

UI Engineering Activities
Use Case-Driven Requirements Analysis
Use Case
System Analysis
Functional Requirements
Use Case- Driven Design and Development
User Interface Prototyping
Software Design, Code and Test
User Feedback
Human Factors Activities
Task Development
UI Conceptualization
User Requirements Analysis
UI Screen Design Support
Usability Testing
Usability Support

Figure 2. A Framework for User-Driven and Use Case-Based for User Interfaces Engineering.

Mainly the framework has been used in 10 projects we conducted at CRIM (Computer Research Institute of Montreal) between 1997 and 2000. All the projects are related to Web-based interactive systems including, for example, an environment for managing ISO 9001 documentation, a tool for sharing resources as well as a Web-based training system. RESPECT and use case-driven approaches were used simultaneously by software and usability experts. At the end of each project, we conduct a series of ethnographic interviews where all participants were interviewed. We asked them to describe their activities during the projects and to highlight the difficulties in term of communication. We also reviewed the framework with all participants and asked them about potential improvements.

## 6 Conclusion and Further Investigations

In this paper, we presented our investigations on how to improve and mediate the communication between usability expert and software development teams. With respect to experimentation, two specific processes constitute the focus of our interests: use case-driven and the user-centered requirements engineering processes. Further to the framework for improving software-to-usability engineering communication we defined, we identified the following principles that we consider as critical issues.

First, the requirements of an interactive system must be defined on two levels, but not independent of one another as it is today. The first level is concerned with the specification of the context of use, and the second focuses on functional requirements. Different specification notations may be used for the two levels, but they should exploit an integrated representation of all the requirements artifacts. In our case, we adopted the text-based forms as used in RESPECT and the graphical representation of use cases as defined in Unified Method Language.

Secondly, the list of artifacts describing the context of use ensures a good usability specification. Better still, this list can assist with generating functional requirements, at least to a limited extent. This result is fundamental because it can minimize requirements artifacts inconsistency and improve communication between software and usability engineers.

## References

Artim J., VanHarmelen M., and al.; “Incorporating Work, Process and Task Analysis into Commercial and Industrial Object-Oriented System Development”, *SIGCHI Bulletin,* 30(4), 1998.

Booch G., Jacobson I., and Rumbaugh J.; *Unified Software Development Process*. Reading, Massachusetts: Addison-Wesley, 1999.

Cockburn A.; “Structuring Use Cases with Goals.” Journal of Object-Oriented Programming, Sept-Oct 1997 and Nov-Dec 1997.

Constantine L.L., and Lockwood L.A.D.; *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Reading, Massachusetts: Addison-Wesley, 1999.

Forbrig P.; "Task and Object-Oriented Development of Interactive Systems – How many models are necessary?" *Design Specification and Verification of Interactive Systems Workshop DSVIS'99*, Braga, Portugal, June 1999.

ISO 13407 Standard: Human Centered Design Processes for Interactive Systems, 1999.

Jacobson I.; *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley Object Technology Series, Reading, Massachusetts: Addison Wesley, 1994.

Jarke M.; "Scenarios for Modeling," *Communications of the ACM* 42(1), 1999.

Kaindl H.; "Combining goals and functional requirements in a scenario-based design process". In *People and Computers XIII, Proceedings of Human Computer Interaction '98* (HCI'98), pages 101–121, Sheffield, UK, September 1998. Springer.

Krutchen P.; "Use Case Storyboards in the Rational Unified Process," Workshop on Integrating Human Factors in Use Case and OO Methods. 12th European Conference on Object-Oriented Programming. Lisbon, Portugal, June 14-20, 1999.

Maguire M.C.; *RESPECT User Centered Requirements Handbook*, EC Telematics Applications Program, Project TE 2010 RESPECT, Wp5 Deliverable D5.3, 1998.

Nunes N.; "A bridge too far: Can UML finally help bridge the gap?" *Proceedings of INTERACT'99 Doctoral Consortium*, Edinburgh, Scotland, 1999.

Ralyte J.; "Reusing Scenario Based Approaches in Requirement Engineering Methods: CREWS Method Base". In *Proceedings of the First International Workshop on the Requirements Engineering Process - Innovative Techniques, Models, Tools to support the RE Process*, Florence, Italy, September 1999.

Rosson M.B.; "Integrating Development of Tasks and Object Models." *Communications of the ACM*, 42(1) 1999.

Seffah A., and Hayne C.; "Integrating Human Factors in Use Case and OO Methods," *12th European Conference on Object-Oriented Programming Workshop Reader*, Lecture Notes in Computer Science 1743, 1999.

# An Evaluation Framework for Assessing the Capability of Human-Computer Interaction Methods In Support of the Development of Interactive Systems

**Yousef Daabaj**

Information Systems Institute,
The University of Salford, Manchester, M5 4WT, UK

Y.H.Daabaj1@salford.ac.uk

**Abstract:** In the course of this research project the output of four Task Analysis (TA) methods were investigated, explored and evaluated to ascertain whether they could support the Requirements Analysis (RA) phase and so contribute directly to other activities in the development life cycle for Interactive MultiMedia (IMM) systems. The research discusses the success and failure factors particular TA methods. The problems of an IMM systems development life cycle are linked to the weaknesses of the Requirements Analysis phase and in particular to the incomplete support of TA methods and techniques used within the Requirements Analysis phase. The outputs of the selected TA methods are evaluated according to four factors, which are represented as an Evaluation Framework (EF). Each factor represents specific criteria and features that TA methods should cover in their processes and outputs. The findings show that TA methods have a number of weaknesses in the support of and the contributions they make. Therefore questions and recommendations are considered about how the methods can be improved in order to obtain better requirements.

**Keywords:** Task analysis, requirements analysis, human-computer interaction, interactive systems, world wide web, multimedia.

## 1 Introduction

The impetus for the research study came from the growing importance of Task Analysis (TA) methods in the success of the overall IMM systems design and development life cycle. This was coupled with the perceived limitations and insufficient contributions of Task Analysis methods to support the Requirements Analysis phase in order to capture the new requirements, distinctive features and characteristics, domain, context and environment of IMM systems. The initial aim of the research was to investigate, explore and evaluate existing TA methods for use and application within the IMM domain and environment, in terms of both theoretical background and practical application. The objective of this research was, therefore, to evaluate the capability of TA methods. The nature of Task Analysis is such that it was felt to be most appropriate to do this in an applied context. Thus, in this project a variety of TA methods have been used to assess the adequacy of a proposed design for a World Wide Web (WWW) site/systems within a particular IMM context. The domain and environment chosen was one which will help a group of research students conduct their doctoral programme as carried out at the University of Salford, UK. This was because the range of problems that arise in using TA methods often only becomes evident when the methods are used in anger. Thus in order to allow a realistic examination of TA methods, the results of the application of these methods, together with their input into the later design activities, have been analysed and compared, both to each other and to a defined schema with its set of special characteristics and criteria. These characteristics and criteria have been represented within a framework that consists of four factors. The purpose of the framework was to assess the capability of TA methods, and to consider Task Analysis in a general way within the IMM context. In this way we were seeking to provide general guidelines, as well as identify aspects and issues to be considered in TA methods themselves, with regard to what they can offer and what they might contain in their final analysis output.

Most of the descriptive efforts in HCI have focused on developing user and task models that can be used to analyse, predict, or explain the performance of users with different interface and system designs, but do these models really provide the necessary descriptive capabilities? The question addressed by this research is simple in its expression but daunting in the work required to answer it. The main question that this research effort is trying to answer is:

*Can the output of Task Analysis (TA) methods support and contribute directly to the Requirement Analysis (RA) phase of the development life cycle of IMM systems?*

In order to understand the relative contributions of each method and technique to the success or otherwise of a design, one needs to know how well each fare in real design settings. The main question is related to a number of sub-questions that have to be answered, and these are:

- How does IMM systems design and development differ from that of traditional software systems with respect to the Requirements Analysis process?
- Why do current IMM information systems require a frequent need to redesign the system?
- How is the Requirements Analysis activity, and in particular Task Analysis, a very important process in designing IMM information systems?
- What can and does each Task Analysis method and technique describe and contribute?
- What decisions and analyses remain outside the scope of any method or technique?

## 2 Background and Motivation

The current emphasis on user-centred design for interactive technologies (Martijn et al, 1999; Annett, 2000) places great emphasis on understanding the user in attempting to develop more usable artifacts. To this end, design teams are urged to perform user and task analysis at the earliest stages of product development and to consider the nature of the users' cognitive and physical pre-dispositions and abilities. These user characteristics are correctly seen as important in constraining the available design options and, if attended to, increasing the likelihood of producing a usable application.

The 1990s have witnessed the world-wide development and utilisation of Interactive MultiMedia (IMM) Systems, and the development processes for such systems have shown a number peculiarities that differentiate them from so-called “classic” software development activities. It is already clear that designers of IMM systems should be made aware of these specific features, constraints and peculiarities as early as possible in the development process. Although IMM design is related to traditional software design, many of its aspects are arguably different from those applicable to sequential media and computer-based instruction, as well as from hypertext (Lowe, 1999). The traditional software life cycles provided structure to the development of large software systems that were mainly concerned with data-processing applications in business. These systems were not highly interactive. Consequently, issues concerning usability from an end-user’s perspective were so that important (Dix et al, 1998). IMM is inherently multidisciplinary in nature and differs in many aspects from the traditional software development. Developing a usable IMM system involves a complex set of design activities and processes. However, it is widely recognised that the design of a product can only be as good as the statement of requirements for that product. This means striving for a requirements specification that is as unambiguous, complete and consistent as possible, so that progress towards the goals specified can be verified in the course of the design. Therefore, future IMM systems will have to be carefully matched to the work environment, expectations, characteristics and tasks of the target users if they are to be successful. This in turn suggests that such issues will become progressively more central to the life cycle of IMM systems’ design and development.

Since the aim of Human-Computer Interaction (HCI) is to produce designs that fit their expected context of use, in order to support people so that they can carry out their activities productively and safely, it is becoming one of the most important issues in information systems development. Diaper (1989) describes the goal of HCI as being to “develop or improve the safety, utility, effectiveness, efficiency and usability of system that include computers”. The study of HCI helps to determine how this computer technology can be made more user friendly (Smith, 2000). These goals are extremely important and relevant in considering IMM systems to support learning. HCI research and design practice has long recognised that IMM systems are of little value if they do not support users in performing their work tasks (Hamilton, 1999). HCI is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them (ACM, 1992). The relevance of HCI is rapidly increasing, giving rise to a growing number of users with a wide range of skill levels making greater demands upon IMM systems in a wide variety of contexts. The discipline of HCI is now widely recognised as a valid partner in design methodology, and the concept of Task Analysis (TA) is often considered of central importance, although the exact meaning of the label varies widely depending on the design approach (Van der ver, 1996).

Given the increasing prominence of the role of TA and user modelling in systems design, the motivation for the research centres on the application and use of a number of TA methods in supporting the analysis and design of usable IMM systems. Its contribution is, therefore, to the emergent disciplines of HCI, Requirements Analysis (RA), Task Analysis (TA) and Interactive Multimedia (IMM) information systems. The author has chosen to study of the activities involved in Requirement Analysis with particular attention to the use of TA methods. This is because Requirements Analysis is a crucial area for the success of the whole IMM development process, since it is the only way to ensure that the final system will be appropriate for the users and their needs. In order to make TA methods capable enough to support the design and development of IMM systems, their context-in-use needs to be actively and effectively exploited. As such, the focus here will be on the use of TA methods to analyse the requirements for a proposed World Wide Web (WWW) site within an IMM environment. The WWW system was selected to provide a complex and interactive environment, context and domain that would help to explore and evaluate the support, contributions and capability of chosen TA methods.

## 3 Problems of the Early Analysis Process

A profusion of notations, methodologies and techniques is documented in the literature of HCI, for example (Diaper, 1997; Lim, 1994; Johnson, 1992). Many of these approaches have been used as techniques during the Requirements Analysis (RA) activity of the interactive systems development process in order to provide descriptive models of the work-task knowledge that people possess. From the literature review it was clear that most Task Analysis (TA) techniques represent a serious attempt by HCI researchers to help designers to develop more usable systems. It has been generally accepted that Task Analysis may substantially contribute to the design of usable products because it focuses specifically on the end user. Task Analysis investigates users' characteristics and the task world of which they are a part, and the information gathered should be recorded in a task model that captures the relevant aspects. The literature on requirements engineering contains little in the way of either theoretical guidance or empirical case studies relating to the specification of requirements for IMM information systems (Jones, 1996). The lack of capability of HCI techniques (in particular TA methods) means that the development costs are often high, and that the quality and usability of the resulting systems are frequently low, with the capabilities of new technologies being only poorly exploited. Although specifying the requirements of the new system to be built is one of the most important parts of the life cycle of any project, its support in practice is still insufficient (Davis, 1993).

The emphasis in systems development has traditionally been on building systems that meet specific functional requirements, without a sufficiently detailed understanding of the cognitive and physical capabilities and expectations of the intended users, or a clear view of the context within which the system will be used. In addition, although basing initial concept design on an understanding of users and their tasks has been advocated for some years, confusion still reigns as to what this means in actual practice. Some authors claim that the required clarity resides in traditional approaches to TA (Dix et al, 1998), but not only are these approaches infrequently used (Bellotti, 1990; Lim, 1996), other researchers question their adequacy and usability (Preece, 1994). The biggest problem is often not in simply applying the technique, but in communicating the results produced to a client or to other members of a design team. If the results of Task Analysis are not communicated well, then their value to the development process rapidly diminishes. The criticism of current TA techniques is that they do not specify how multimedia development may be taken through from analysis specification to detailed design specification. Existing TA techniques need to capture:

- the requirements in their original format (e.g., graphics, text, audio, video),
- the context of the interaction and interactivity style and level,
- the context of the stored information content and type,
- the delivery environment and technical constraints,
- the corresponding limitations that should be placed on multimedia output.
- the navigational structure of the system,

The refinement of these original requirements and the identification attributes associated with the requirements are outside the scope of most current methods, and therefore when they are dealt with it is often in an informal way.

## 4 The Role of Task Analysis (TA)

Human-Computer Interaction (HCI) has a role in the design and development of all kinds of systems, ranging from those like air traffic control and nuclear processing, where safety is extremely important, to office systems, where productivity and job satisfaction are paramount, to computer games, which must excite and engage users (Preece, 1993). An information system will be of no value if it does not contribute to the improvement of the work situation for people in the organization. Therefore, it is not enough to study the contents of the information system. The activities people perform in an organisation and how these could somehow be improved must also be examined. The fundamental idea of Task Analysis (TA) lies in a science-based and purpose-oriented method or procedure to determine what kind of elements the respective task is composed of, how these elements are arranged and structured in a logical, or/and timely order, how the existence of a task can be explained or justified, what the driving force to generate it was, and how the task or its elements can be aggregated to another entity, composition, or compound. Task

Analysis could therefore be a central activity in system design. Task Analysis helps ensure that human performance requirements match users' needs and capabilities and that the system can be operated in a safe and efficient manner. As technical systems become more sophisticated and pressure to reduce manpower in them increases, there is a severe risk that unique human skills and abilities may not be used as effectively as they should, thus degrading the potential performance of a system. Therefore, TA as one of the main analysis techniques for human-machine systems design plays an important role in different project development phases.

A critical issue in the emerging area of IMM Information Systems is the ability of these systems to fulfil the information requirements of various user groups. The importance of the TA process is rapidly increasing with the growing number of users with a wide range of skill levels making greater demands upon IMM systems in a wide variety of contexts. Task Analysis is about examining the context and criteria in order to establish a solution, as well as about examining the context and criteria associated with goals in practical situations to identify how they are carried out or what problems are associated with their execution. Therefore, TA is very useful to the extent that it helps us to improve the design or implementation of systems or, at least, to focus upon areas of poor human performance. IMM systems are designed to fulfil particular goals and should aid people to accomplish them. The way people accomplish goals is by executing tasks. Therefore, a task model can reveal much about the way in which tasks need to be organised within an IMM system. Therefore, a good picture of human task performance is very important for the design of IMM systems. Table 1 shows the relation between TA and RA. Task Analysis methods are expected to indicate, for instance:

- which tasks will be used often or only infrequently,
- to allocate tasks between multimedia system and the target user,
- which media or combination of media the designer must pay attention to,
- the main features of the interface and the structure of the content of the system, etc.

Both RA & TA have elements (concepts) in common which is to produce a definition:

- that can be used as a basis for realisation of a software system that delivers the behaviours and features required.
- that is sufficient as validation criterion for the system produced.
- to establish constraints to support later design decisions and extensions.

| **RA vs. TA** | **Requirements Analysis (RA) [abstract, partial task elements]** | **Task Analysis (TA) [real, complete, representative tasks]** |
|---|---|---|
| **Differ on "who/what" it is about** | RA about system's needs, functions and specifications. | TA about real users and real tasks they want to do. |
| **Purpose/ Aim** | RA specifies *WHAT* the system should do. | TA provides information about *HOW* it should be done. TA is concerned with how a user performs things "tasks". |
| **General process activities** | In general : 1.Requirements identification 2.Identification of software devel-opment con straints 3.Requirements analysis process 4.Requirements representation, 5.Development of acceptance criteria and procedures. | In general: 1.Information collection, 2.Task description, 3.Task Analysis, 4.Representation, 5.Application. |
| **Data collection techniques used** | Observations, Interviews, Focus group discussion, Existing documentation, Checklist, Questionnaires, Videotape, Survey, and TA techniques. | Observations, Interviews, Focus group discussion, Existing documentation, Checklist, Questionnaires, Videotape, Task allocation. |
| **Particular strengths** | Generate a great deal of background information about: problem domain; user requirements and characteristics on the current situation. | Understanding current work in depth, provide a representation of how users perform their tasks, envisionment of how user's work might be in the future, and establishing basis for satisfying jobs. |
| **Similar concepts in different models** | Waterfall model | Iterative design |
| **Expertise or skills required** | Purpose of analysis, Interviewing, Planning and Subject handling, user – analysts communication techniques, Group chairing, Survey design and analysis skills and expertise. | A depth knowledge of TA technique(s), Interviewing, Planning and Subject handling, System knowledge, Purpose of analysis, User – analysts communication techniques, Group chairing, Experience and skills in conducting the analysis. |
| **Differ on "who" does** | Done by systems analyst | Mostly done by User Interface designers, and Human Factors specialist. |

**Table 1:** Requirements analysis vs. task analysis.

## 5 The Principle Behind the Framework

The bottleneck in producing such IMM systems is no longer in the technical stages of building it, but in the preliminary analysis phase of specifying the essence of design (producing the logical structure of the system) and capturing the specific features and characteristics. The advent of larger and more complex IMM systems has resulted in the need to reconsider the ways in which requirements are captured, analysed, formalised, modelled, represented and communicated pertaining to those systems. This reflects the author's view that conventional systems analysis, design and development methods do not cater for IMM systems, and most published work on IMM focuses either on programming or hardware issues (and the real technical challenges involved). The specification of information requirements is a particularly important part of the Requirement Analysis (RA) phase, since poor specification leads to poorly-designed systems which in turn leads to reduced usability of the systems.
The problems of current software design methods highlights the importance of the Requirements Analysis phase in system development, and draws out the importance of integrating and combining with Task Analysis (TA) techniques in order to design usable IMM systems, see Figure 1.

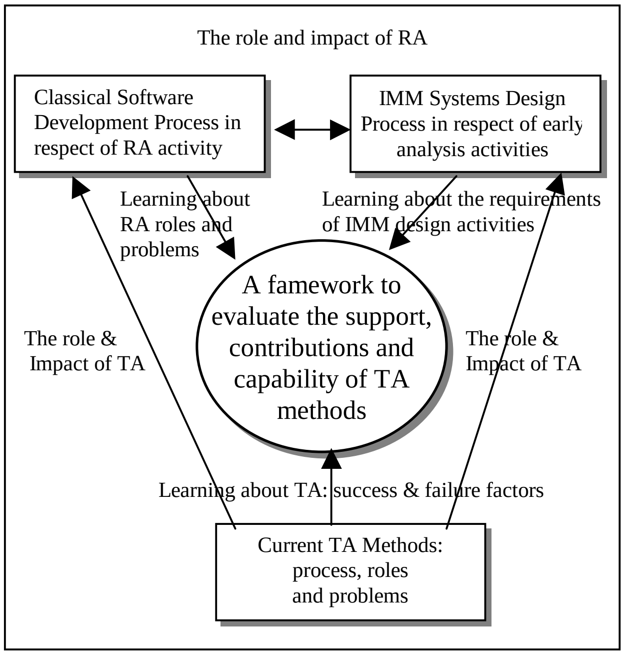

The role and impact of RA
Classical Software Development Process in respect of RA activity
IMM Systems Design Process in respect of early analysis activities
Learning about RA roles and problems
Learning about the requirements of IMM design activities
A famework to evaluate the support, contributions and capability of TA methods
The role & Impact of TA
The role & Impact of TA
Learning about TA: success & failure factors
Current TA Methods: process, roles and problems

**Figure 1:** An overview of the principle behind the evaluation framework.

There is very little published on how to capture, analyse, model, represent and communicate requirements for and design IMM systems (Jones, 1996). IMM system design requires techniques to support the early phase of the development process that have sufficient expressive power to capture the nature of the context of use that supports the multimedia development life cycle. The development of multimedia application places many demands on the multimedia author, including the following:

- a knowledge of the information content required in the application.
- a knowledge of the application user and the requirements of the application user.
- a knowledge of the target user's tasks.
- a knowledge of the working environment within which the system will be used.

## 6 An Evaluation Framework for Assessing the Capability of TA

Although some researchers, for example (Card et al, 1983; Johnson, 1990) have made claims about what TA, HCI principles and methods might contribute to system design, they have aimed at finding desirable criteria and identifying some distinguishing features for assessing the suitability and applicability of TA Methods, for example (Diaper, 1989; Bellotti, 1990; Wilson, 1993) order to support comparisons between them. Finding the right criteria to determine what makes a good TA is difficult to find in the literature, despite the many suggestions that have been made about how TA should be done. It is an open question, as to which of the TA methods will prove most successful to design more usable systems. The views expressed in the literature have identified desirable criteria that should exist within the process/products of current TA models. These criteria were represented in a framework, called "An Evaluation Framework (EF) for Assessing the Capability of TA Methods", which consisted of four main factors. These desirable criteria of each factor aimed to improve the capability of current TA methods in order to support and contribute directly into the Requirements Analysis (RA) phase, which could result in producing better requirements.

**The Scope of Analysis Factor:**

- Requirements Classification
- Identify Task Characteristics and Procedures
- Environmental Characteristics and constraints
- Select and Match Media to Content Analysis
- System Navigational Structure and Access Techniques
- Identify Special Interactions and Features of IMM Systems
- Requirements Implementation Plan

**Representation Form and Support Factor:**

- Variety of Representation Form Provided by TA Method
- The Completeness of Notational Support
- Automated Support to Represent the Features of IMM Systems

**Requirements Mapping Factor:**

- Feasibility Study/ Project Planning/ Problem Definition
- Requirements Analysis & Specification (RA&S)

- Information Design-Content Selection and Organising
- Choosing the Correct Navigational Structure and Access Techniques
- Choosing: Design Approach and User Interface Approach
- Iterative & Prototype Process
- Implementation
- Web/System Testing & Delivery

**Core Criteria Factor:**

- Understandability of Requirements Output
- Correctness of Requirements Output
- Usability
- Validity

In order to explore the reviewed TA methods' support, contributions and capabilities, as well as to assess how their capability could be improved, the Evaluation Framework relates the scope of TA methods, in particular to the RA phase (i.e., to obtain better requirements), and in general to the IMM systems development process. The concern was with the use of TA methods to support the RA activities, see Figure 2 below.

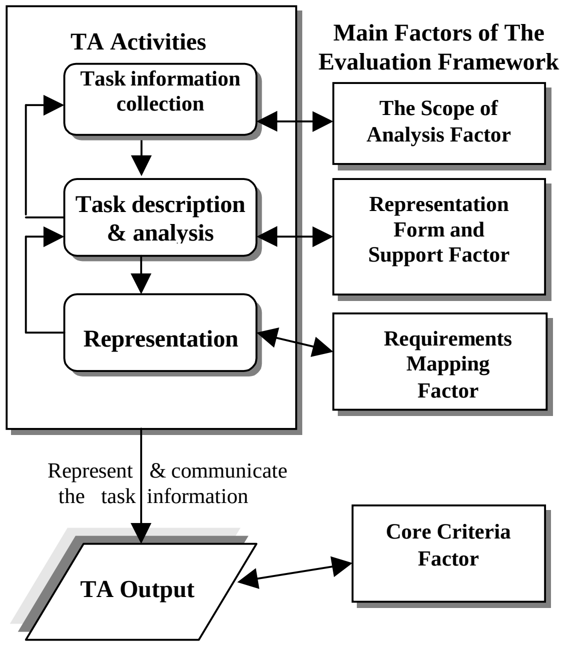

TA Activities
Task information collection
Task description & analysis
Representation
Main Factors of The Evaluation Framework
The Scope of Analysis Factor
Representation Form and Support Factor
Requirements Mapping Factor
Represent the task
& communicate information
TA Output
Core Criteria Factor

**Figure 2:** An Overview of the Evaluation Framework.

To make a useful evaluation of TA methods, is to decide whether they are capable enough to support and contribute directly to analysis and web site design within the IMM context and environment. This requires a scheme that asks what ought to be in a TA product, what can/cannot be offered, what the role and impact of these methods is in designing the usability of IMM systems, what the scope of the potential contributions is (i.e., what is the usefulness of the analysis), what resources (skills, time, modelling tools) would be required for application of the method (i.e., how usable is the method), and whether information obtained leads to positive design recommendations. The framework is a conceptual representation in tabular format and textual form, and is aimed to structure and understand a range of concepts of Task Analysis methods. Each factor represents some specific issues, desirable features and elements for the RA phase and other activities of the IMM approach, the TA method should cover and represent these features somehow in their process and outputs. Clearly these kinds of "criteria" are qualitative and not quantitative.

## 7 The Case Study

Arguably the most critical activity/phase in the development of a large, interactive and complex multimedia information systems is to capture, analyse, organise, represent and communicate the output of the early analysis process (Wood, 1998; Lowe, 1999). Furthermore, there is currently a growing interest in the explicit introduction of Task Analysis (Annett, 2000; Lim 1996). This has come about as a result of the realisation that the development of friendly IMM information systems is becoming increasingly difficult as contextual factors become broader and more sophisticated (Dix et al, 1998). The researcher has to be domain-literate in order to understand the significance of events and activities in their domain context. It is argued that the nature of information collection processes under a naturalistic setting must be opportunistic rather than systematic.

Case study is a method for doing research which involves an empirical investigation of a particular contemporary phenomenon within its real-life context using multiple sources of evidence. A case study can describe a phenomenon, can build theory, or can test existing theoretical concepts and relationships (Cavaye, 1996). To complement the theoretical assessment of Task Analysis (TA) methods and their evaluation against the framework (defined factors with their specific criteria), an empirical evaluation of TA methods for analysing user information requirements and tasks should be implemented. Although experimental laboratory studies could be used to study specific aspects of the task, it was felt that within the time span and scope of the research empirical study would allow consideration of a wider scope of TA issues and problems. The research is interested in human activity in a scientific approach (Positivist – Case study). At a more theoretical level, practical uses of TA help to further ergonomics knowledge and theory in this area. Through practical use, problems or deficiencies in methods will be highlighted; methods can then be improved to increase their capability. When a researcher selects case study as an appropriate method for a research study, the strengths of case study are considered of importance and the weaknesses are accepted as method-related limitations of the research (Cavaye, 1996).

## 7.1 Aim and purpose of the case study

Task Analysis covers a wide variety of roles, goals, methods and techniques, and because of this, it is useful to specify at the outset what is meant by a task. The task focused on here is that which must be performed from the point of view of Ph.D. research students, who usually carry it out, so that the potential user (Ph.D. student) of the system may be taken into account during the design process. The main goal of the analysis is to describe sets of tasks, in terms of tasks which must be executed, and that may be useful for the design of a WWW site/system. The aim therefore is to describe the execution of a set of tasks, as they are perceived by Ph.D. research students who perform them: in other words, how they would explain the performing of these tasks to a beginner. However, the basic idea behind the case study was based on using the selected TA methods in order to find out how Ph.D. research students conduct and plan their doctoral research programme from the start to the final step of writing up the thesis. The results of using TA methods were collected and analysed, compared and evaluated as to whether their outputs were capable enough to support the Requirement Analysis phase. It was planned in the future to use the results of this research (i.e., the analysis output) to design and develop a WWW Web sit in order to teach and introduce the basic steps, concepts, terms and aspects of all details of how Ph.D. research students can conduct their doctoral research, as well as to provide basic information that Ph.D. research students may find useful, to avoid the problems they are currently facing. It could save a great deal of time and frustration if research students get to know these basic procedures.

The research questions and objectives show that four-case designs (i.e., four examples of Task Analysis methods) are more desirable than a single-case study, because they allow us to examine the boundaries of TA methods in more detail, they also allow for cross-case analysis and for the extension of the boundaries of current Task Analysis approaches. More than one case may yield more general research results and enable the researcher to relate differences in context to constants in process and outcome. The Task Analysis methods that were selected, reviewed, explored, used, applied and evaluated during the case study were (four):

- The First Case: Analysis and Training in Information Technology Tasks: Hierarchical Task Analysis (HTA) method, (Shepherd, 1989).
- The Second Case: Task-Action Grammar (TAG) method: The Model and it's Developments, (Payne, 1989).
- The Third Case: Supporting System Design by Analysing Current Task Knowledge; Task Knowledge Structures (TKS) method, (Johnson, 1992).
- The Fourth Case: Task Analysis for Knowledge Descriptions (TAKD) method: The Method and an Example, (Diaper, 1989).

For each TA method, only one reference was used to obtain the details of the product (the references are those cited above). This is intended to avoid the problem of alternative versions of a TA being obtained from different references.

The purpose of the case study was to provide a complementary input into the study, and to offer an appropriate context for the study of the methods to complement the theoretical considerations of analysis. The aim of the case study, however, was to demonstrate how Task Analysis methods can be applied within an IMM environment and to evaluate and explore the limitations and boundaries of the process of the selected Task Analysis methods. The aim is also to find out the critical failure factors that limited the scope of analysis, representation form, usability, mapping and contribution of specifications to other development activities.

## 7.2 The problem situation

Doing doctoral research involves steps, skills, knowledge, planning, scheduling, etc, required by Ph.D. research students. Why should students know how to do research? One reason is that they are studying for a degree, which requires a thesis. By knowing how to begin and how to conduct the research successfully, they would save time, money and maintain more control over the research by discovering what kinds of things they need to know and what kind of help they need. Another reason for knowing how to do research is that they will be better able to weigh the value of other people's research. The purposes of describing the procedures for the progression of doctoral research towards their final degree are: to encourage the most qualified and able students to continue in the doctoral programme and to assure their steady progress toward completion of the Ph.D. degree without imposing onerous burdens; to protect those students who are unlikely to succeed in the programme from pointless investment of time and effort and to help maintain and promote the high quality of the Ph.D. doctoral research programme.

A problem situation was selected in order to provide contexts and applications of TA method where adequate access (i.e., easy access to the expected users-Ph.D. research students) could be granted and where the availability of the application domain to analyse existed. Task Analysis methods considered typical decisions that had to be made and examined what information needed to be considered by the Ph.D. doctoral research students to conduct their research stages and also what kind of problems needed to be avoided.

### 7.3 Sources of information

A clear description of data sources and the way they contribute to the findings of the research is an important aspect of the reliability and vitality of the findings. The case study's unique strength is its ability to deal with a full variety of evidence-documents, artifacts, interviews, and observations. The goal of using different data collection is to obtain a rich set of data surrounding TA processes, usability and capability, boundaries, limitations and the possible causes of failure, as well as capturing the contextual complexity. The data to be collected will depend on the research questions and the unit of analysis.

The more precise the goals of the investigation, the more specialised the data collection can be. Therefore techniques applied will depend not only on the methods of TA that are to be used, but also on the goal, the purpose of the study and the time and resources available. This is based on the view that different data collection techniques can offer a different perspective on the task. For example, document studies and interviews with real users determine what should be done and what they think they have to do. For an analysis of all but a simple task it is likely that several methods will be employed to collect the task data, either independently or in conjunction with one another. Ainsworth (1995) report that in order to ensure that TA is reliable, it is useful to use different sources of information while developing and rechecking the TA. The resulting TA can only be as good as the original data (Dix et al, 1998).

Whilst the TA methods provide a framework for organising the task descriptions and information, the quality of the analysis depends on the information input into the analysis. Therefore, it is important to select a data collection method that provides information in a format that can be used by the analysts. Though observation and interviews are most frequently used in case study method, methods of collecting information were selected which were appropriate for the task, that is, in respect to the research method (i.e., Positivist - Case study), questions and hypothesis. However, in order to find out precisely what the task entailed and where the problem areas lay, the requirements information used in the case study was obtained and collected from a variety of diverse sources, and was governed entirely by the availability of documents and access to Ph.D. research students. Relevant task domain information had to be collected focusing on different phenomena, and using different methods of data collection. Based on an analysis of the character of the knowledge sources in the reviewed TA frameworks, different methods were identified to collect all information needed to construct the required task models. The techniques addressed the students' research procedures and identified the main tasks, stages, goals and reasons for particular task structuring. The data collection techniques which were used are as follows:

- structured and un-structured interviews (subject-based);
- questionnaires (subject-based);
- documentation;
- walk through,
- focus group discussion.

# 8 Research Reflections

A major problem with the introduction of information systems is the fact that a very large proportion of them fail to meet their initial aspirations (Smith, 2000). This is due to the lack of obtaining better requirements during the Requirements Analysis (RA) phase as a result of inadequate support and contribution by Task Analysis (TA) methods and techniques. The followings are what the author has learned from the research process:

- ♦ advances in information technology and the expansion of the numbers and types of systems and applications that use the new technology in innovative ways are quickly leading to a world in which pervasive computing is the norm. The result impacts upon people in both individual and organisational contexts and in society at large. This is particularly so with recent developments of the internet and web-based applications and with the rapidly-expanding range of technological devices that are being embedded in many products and systems. It is essential that the research, tools, methods and techniques developed in the field of HCI are considered and integrated into the development of software and systems using these new technological advances, if the anticipated benefits are to be fully realised.
- ♦ there are many tools, methods and techniques that can be used to gather task-related information and place it into a meaningful context. Each of these has its particular strengths and weaknesses. Therefore, the success or otherwise of any TA exercise will to a large extent depend upon how the mix of TA techniques and data sources is selected and applied. Ideally, for any TA project there should be an overall plan and subsequent close monitoring to ensure that objectives are met.
- ♦ mechanisms for collecting the raw data could similarly be structured. The analysis of existing

tasks tends to rely largely upon observation, interviews and documentation. While techniques for analysing these data independently are available, these have yet to be integrated and combined with the task analytic method. Conceptually, at least, this is not a difficult undertaking, but one that has yet to be proven in the field.

- ♦ at present, little is known of the reliability and validity of current TA methods. There is certainly plenty of scope for more research of this kind to be conducted, and the development of methods would clearly be beneficial. In addition, standardised training and documentation of methods should be developed, as acquisition appears to rely largely upon self-development at present.
- ♦ the way in which all user views, tasks, needs, requirements, preferences, characteristics, and environment (i.e., physical, aspirational and functional) are re-ascertained will be a major factor in supporting the Requirements Analysis (RA) phase which leads to obtaining better requirements, as well as in determining the quality (effectiveness, efficiency and satisfaction) within the end product.
- ♦ since Task Analysis is a time-consuming activity, a guidance to the application of the method, in particular during the generification, can limit the employment of resources necessary to carry out the complete TA process. Essentially, it is apparent from the case study that there is a need for the development of guidelines to address the various stages of undertaking a TA, including planning the methodology, data collection, data analysis, presentation of TA information and mapping the output information into each activity of the development life cycle. However, such guidelines should provide systematic advice, rather than inflexible, prescriptive rules.

## 9 Conclusion

Attempts to tie the various theoretical and methodological strands together will only come about by framing the problem in terms of human activity in context. This ecological perspective seems to be fundamental to task analysis (Annett, 2000). In this research the application of TA methods has been used to assess the adequacy of a proposed design for a World Wide Web (WWW) site within an IMM environment which will help research students conduct and deliver their doctoral programme as carried out at the University of Salford, UK. The results of the application of TA methods and their input into the design activities were analysed and compared both to each other and to a defined framework (a set of four main factors with desirable criteria that should exist in the output of the TA process) in order to evaluate their capabilities. These criteria were developed as a result of an extensive literature review, where methods which could potentially be used for this purpose were identified. The findings however, have shown that TA methods have a number of weaknesses in the support and contribution they made.

The philosophy of IMM is quite unlike that of the more traditional mediated systems which are designed for a specific task or range of tasks. IMM systems have special and different characteristics, which make their design and development process more difficult and thus should be approached differently from the early phase. Some of these characteristics are: the complexity of the system’s navigational structure, media-selection, integration and synchronisation, the type and variety of contents, the style and the level of the interactivity and interactions and the UI design, etc. The research argues that the design process can be more efficient or optimised by the continuous use and application of the relevant TA methods throughout the development cycle in order to aid IMM system design activities.

Surprisingly little Task Analysis (TA) has appeared for one of the most discussed and fastest-growing interactive computer application the ‘WWW’. The reviewed TA methods did not however include any concepts explicitly intended for interactive systems involving multiple media types. This was an intentionally important aspect of our chosen application (i.e., WWW). This failure of the reviewed TA methods can be seen as a further example of how the technological developments of IMM pose novel design issues for HCI, in this case raising the need to broaden the scope of TA. The scope of the selected TA methods needs to be extended to cover and focus on the complete system’s design and development activities, including for example the application domain of the intended system (i.e., the major constraints on design decisions) and to bridge the gap that exists between the analysis process and the subsequent development activities. Proper operation of any TA method should form part of a general planned approach to a problem.

## References

ACM SIGCH (1992), *Human-Computer Interaction*, Web Site-Available at http://www.acm.org/sigchi/cdg/cdg2.html/

Ainsworth, L. & Pendlebury, G. (1995), Task-Based Contributions to the Design and Assessment of the

Man-Machine Interfaces for a Pressurized Water Reactor, *Ergonomics* 38(3), pp. 462-474.

Annett, J. & Stanton, N.A. (2000), *Task Analysis*, Taylor & Francis, London & New York.

Bellotti, V. (1990), A Framework for Assessing the Applicability of HCI Techniques. *In Proceedings of INTERACT'90*, Cambridge, North-Holland, Amsterdam, 213-218.

Card, S.K., Moran, T.P. & Newell, A. (1983), *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates.

Cavaye, A.L.M. (1996), Case Study Research: A Multi-Faced Research Approach for IS. *Information Systems Journal* 6, Blackwell Science Ltd., 227-242.

Davis, A.M. (1993), *Software Requirements: Objects, Functions, and States*. Prentice-Hall, Englewood Cliffs, NJ.

Diaper, D. (1989), *Task Analysis for Human-Computer Interaction*. Ellis Horwood Limited.

Diaper, D. (1997), Integrating HCI and Software Engineering Requirements Analysis: A Demonstration of Task Analysis Supporting Entity Modelling HCI and Requirements Engineering. *ACM SIGCHI Bulletin* 29(1), 41-50.

Dix, A., Finlay, J., Abowd, G. & Beale, R. (1998), *Human-Computer Interaction*. Second Edition, Prentice Hall Europe, ISBN 0-13-239864-8.

Hamilton, F., Johnson, H. & Johnson, P. (1999), PRIDE: Task-Related Principles For User Interface Design, *in* M.A. Sasse & C. Johnson (eds.), *Human-Computer Interaction – INTERACT '99: Proceedings of the Seventh IFIP Conference on Human-Computer Interaction*, IOS Press.

Johnson, P. (1992), Task Knowledge Structures (TKS) Theory, and Knowledge Analysis of Tasks (KAT) Method. *In Human-Computer Interaction: Psychology, Task Analysis and Software Engineering*. P. Johnson (ed.), McGraw-Hill, 151-192.

Jones, S. & Britton, C. (1996), Early Elicitation and Definition of Requirements for an Interactive Multimedia Information System. *In the Proceedings of ICSE'96 Conference*, Session 2A, 12-19.

Johnson, H. & Johnson, P. (1990), Designers-identified Requirements for Tools to Support Task Analysis. *In Human-Computer Interaction (INTERACT'90)*, D. Diaper., D. Gilmore., G. Cockton and B. Shackel (eds.), IFIP, Elsevier Science Publishers B.V. (North-Holland) , 259-264.

Lim, K.Y. (1996), Structured Task Analysis: An Instantiation of the MUSE Method for Usability Engineering. *Interacting with Computers* 8(1), 31-50.

Lowe, D. & Hall, W. (1999), *Hypermedia and the Web: An Engineering Approach*, Wiley.

Lim, K.Y. & Long, J. (1994), *The MUSE Method for Usability Engineering*, Cambridge University Press, Cambridge.

Martijn, V.W., Van der Veer, G.C. & Anton, E. (1999), *Breaking Down Usability*, Faculty of Computer Science, Vrije Universiteit Amsterdam.

Payne, S. & Green, T.R.G. (1989), *Task-Action Grammar (TAG) method: The Model and it's Developments*, in Task Analysis for Human-Computer Interaction, Diaper, D. (ed.), Ellis Horwood Limited, Chichester, pp.75-107.

Preece, J. A. (1993), *Guide to Usability: Human Factors in Computing Reading*, MA, Addison-Wesley.

Shepherd, A. (1989), *Analysing and Training in Information Technology Tasks*, in Task Analysis for Human-Computer Interaction, D. Diaper (ed.), Ellis Horwood Limited, Chichester, pp. 15-55.

Smith, A. (2000), *Human-Computer Factors: A Study of Users and Information Systems*, Information Systems Series, The McGraw-Hill Companies.

Van der Veer, G.C., Lenting, B.F. & Bergevoet, B.A.J. (1996), GTA: Groupware Task Analysis - Modelling Complexity, *Acta Psychologica* 91, 297-322.

Wilson, S., Johnson, P., Kelly, C., Cunningham, J., Markopoulos, P. & Beyond, P. (1993), Hacking: A Model-Based Approach to User Interface Design. In *People and Computers VIII*, J.L. Alty, D. Diaper and S. Guest (eds.), *Proceedings of HCI'93*, Cambridge University Press, 215-231.

Wood, L.E. (1998), User Interface Design: Bridging the gap from User Requirements to Design. CRC Press.

# Incorporating Usability into an Object Oriented Development Process

Xavier Ferré
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo
28660 - Boadilla del Monte
Spain
xavier@fi.upm.es

## Abstract

Software development organizations pay and increasing attention to the usability of their software products. HCI (Human-Computer Interaction) techniques are employed profusely in software development, but they are not integrated with the Software Engineering development processes in most of the cases. Use cases stand as a bridge between Software Engineering and HCI, because of their popularity in object-oriented development and because its user-centered nature. Giving use cases an additional user-centered focus we can make our way through object-oriented software development combined with usability techniques. From the numerous approaches to software development we have chosen an iterative, incremental and use-case driven process. We briefly analyze two methods with this approach, the Unified Process and Larman's method, considering their suitability for integration with usability techniques. A generally applicable object-oriented development process with integrated usability techniques is presented, following the approach shared by both methods. Specifically, from Larman's method we stick to its idea of giving priority to the interaction design over the design of the internal part of the system. The proposed process gives advice on the usability techniques to be used in every phase of such joint development process.

## 1 Introduction

Usability is not commonly addressed in software development. It is properly addressed only in projects where there is an explicit interest on usability, and the quality of the system-user interaction is perceived as critical by the software development organization. In this kind of projects usability experts drive the development, using mostly usability-related techniques in the phases previous to coding.

Usability techniques are applied following development processes alternative to the Software Engineering ones, due to the fact that it is not solved yet how to integrate usability techniques into Software Engineering development processes [1][2]. One of the virtues of the HCI field lies on its multidisciplinary essence. This characteristic is at the same time the main obstacle for its integration with Software Engineering: while the HCI foundations come from the disciplines of psychology, sociology, industrial design, graphic design, and so forth; software engineers have a very different approach, a typical engineering approach. Both fields speak a different language and they deal with software development using a different perspective.

Software Engineering has traditionally constructed software systems with development focused on internals, on processing logic and data organization [3]. Consequently, software quality has been identified with issues like efficiency, reliability, reusability or modularity. These are aspects of the system that the user is scarcely aware of. In contrast, the interaction with the user has been sometimes left as a secondary issue [2]. Despite the stated aim of building a software system that satisfies the user, after establishing a closed set of specifications the user is forgotten until the first release of the software product. Usability is sometimes wrongly identified to be a graphical user interface issue, which can be addressed after the main part of the functionality has already been developed [4]. When developers perceive usability in this way, they tend to think that after having constructed the "important" part of the system (the internal part), usability specialists can add a nice user interface in order to make the product usable. This attitude leads to systems where usability problems are very costly to fix when identified.

Usability practitioners, on the other hand, have focused on the user and the way he or she interacts with the system. They employ a set of techniques to better canalize the creative activity of interaction design, and to evaluate its products with real users. Focusing on the creative nature of interaction design, they haven't paid attention to issues central to Software Engineering such as how to make their way of developing systems repeatable and structured, or how to estimate and plan their procedures.

A change can be seen in the attention paid to usability. An increasing number of software development companies are beginning to consider usability as strategic for their business, and they are pursuing the aim of integrating usability practices into their Software Engineering processes. One of the leading journals for software practitioners, IEEE Software, has dedicated its January/February 2001 issue to the role of Usability Engineering in software development. Some proposals for integration ([5][6]) present ad-hoc solutions which have been created for particular software development organizations, but they lack a generic approach to be applied to organizations with different characteristics.

In this paper we propose an object-oriented development process where usability techniques are embedded where appropriate. The approach we propose is generally applicable in the development of interactive software systems.

Considering the large number of existing usability techniques, some of them can be more easily applied from a Software Engineering point of view. These ones have been chosen to be embedded in our proposal for a joint development process. Likewise, among the numerous Software Engineering methods, use-case driven approaches are the closest to a usability perspective. They are considered in the next section. Usability-related activities are accommodated into a use-case driven method and the resulting process is described in section 3. The particular usability techniques chosen to be applied in each phase or activity are detailed in section 4. Finally, conclusions and future directions are presented in section 5.

## 2 Use Cases and Usability

Traditionally, software development has taken a variety of forms, with minor or big differences, but with a common focus: trying to build a system beginning from the inner part of it, the internal structure. A different approach stands out in object-oriented software development: use-case driven development.

Use cases are a user-centered technique in its conception, so they can fit well with the HCI approach. Therefore, use cases seem to be the best starting point for the integration of usability techniques and Software Engineering.

Nevertheless, using the technique of use cases is not a guarantee of a real user-centric development. For that purpose it is crucial that use cases are not automatically converted into technical specifications, in the sense of a production line. When use cases are taken away from the user sphere they loose their main benefit from a usability perspective. That is not to say that technical specifications will not be based in the use cases defined, but they are not the result of some direct use case transformation. Technical specifications reflect the development team decisions, which are taken according to the user needs stated in the use cases. It is important that use cases are not considered as initial design artifacts, because then the modeler can easily cross the line and model them according to a particular internal functionality design.

To get the maximum from use cases as a technique to improve the usability level of the software product, we must complement them with the concept of task used in HCI, specifically in the set of techniques known as Task Analysis ([7]). User intentions and needs must be in the root of the use-case modeling task, in order to have them fixed in the user world all the time, in the context where the system will have to be deployed.

Today's possibly most popular object-oriented method is the Unified Process [8]. This process is labeled by its creators as use-case driven. However, analyzing the role of use cases in the process, it can be observed that the use-case model plays a secondary role compared to system architecture. The use-case model is very important in cycle planning, but once the cycles start use cases are regarded as a preliminary version of elements of the internal functionality design. When design elements are labeled as use-case realizations we are shifting use cases to the design world, and therefore away from the user realm. Overall, we consider that the Unified Process, as described by its authors in [8], is mostly devoted to design, and to a design level very close to implementation. This approach can significantly prevent the development team from adopting a proper user-centered perspective. Yet, we adopt from the Unified Process the iterative and incremental nature of its process, because it allows for early usability testing. Having early in the development cycle a running part of the system can give us the opportunity of testing it with users.

In the object-oriented method proposed by Larman [9] we can also find an iterative, incremental and use-case driven process. This method establishes a distinction between analysis and design that we find useful for our purpose.The analysis phase, while including traditional analysis activities like the creation of a Conceptual Model, it also addresses the design of the interaction of the system with the user, by viewing the system as a black box that receives requests from users and other systems. The external system communication is specified by means of system sequence diagrams and system operation contracts. The details of the internals are left for the design phase. This approach is well suited for an integrated process, as it allows for an interaction design previous to the low-level design that will give form to the inner part of the system.

Unfortunately, despite having a suitable focus, Larman´s method has some important flaws from a usability perspective. On one hand, the Conceptual Model (supposedly an analysis construct) is exploited as a rudimentary database, causing the developer to focus more on database modeling than on gathering user-domain knowledge. On the other hand, an event-driven interaction mode is implicit in its approach. This lack of flexibility regarding the interaction mode can force the production of a software system of lesser quality.

We will take as starting point for our endeavor an iterative, incremental  and use-case driven object-oriented development process. We will supplement this approach with the usability techniques appropriate for each phase of the development, and we will adopt an interpretation of use cases closer to HCI. We consider that such a process can yield useful from both a Software Engineering and a HCI point of view.

## 3 Joint Development Process

We base our process in Larman’s method, but we have adapted his terminology to make explicit the user-centered philosophy. We call *External Design* to the analysis phase in Larman’s method, and *Internal Design* to his design phase. External Design deals with the design of the communication between the outside world and the system, while Internal Design is concerned with the design of the internal structures to give service to that previously designed interaction.

The structure of the process is shown in figure 1, where activities at the same horizontal level can be performed in parallel, and activities which are placed higher inside a phase are performed before than activities placed lower. The process is formed by a preliminary Analysis phase and five phases inside the iterative cycles.

The preliminary phase is as follows:

- **Analysis**: Before taking any decision about the future system, it is defined what the system is supposed to be in general terms (*System Concept Definition*), and what the intended users will be and their characteristics (*User Analysis*). Later on, a more detailed definition of what the system is going to offer to the user is specified (*Task Analysis*). A traditional set of requirements is specified with the addition of particular usability requirements (*Requirements Specification*). To ensure that the set of tasks created suit the user needs, a *Validity and Usability Evaluation* is performed on the requirements and tasks.

The phases performed in every development cycle, for the use cases selected for it, are the following ones:

- **Analysis Refinement**: A deeper understanding of the problem is obtained from the design effort in the previous cycle. The documents and models created in the preliminary analysis are consequently revised in this activity.
- **External Design**: The tasks identified in the Task Analysis are designed more precisely (*Task Design*) in parallel with the definition of interaction objects and their behavior (*Interaction Conceptual Design*). The resulting interaction scheme is evaluated (*Usability Evaluation*).
- **Internal Design**: The classes to support the interaction designed in the previous phase are specified, along with their behavior (*Object Oriented Design*). The graphical user interface that gives shape to the interaction design is built with the contribution of experts in graphic design (*Interaction Visual Design*).
- **Implementation**: The structures designed in the previous phase are taken to a specific programming language, and converted into a working system.
- **Testing**: The system built is subject to tests to ensure that complies with the requirements (*Verification and Validation*). In particular, it is tested with users for compliance with usability requirements (*Usability Evaluation*).

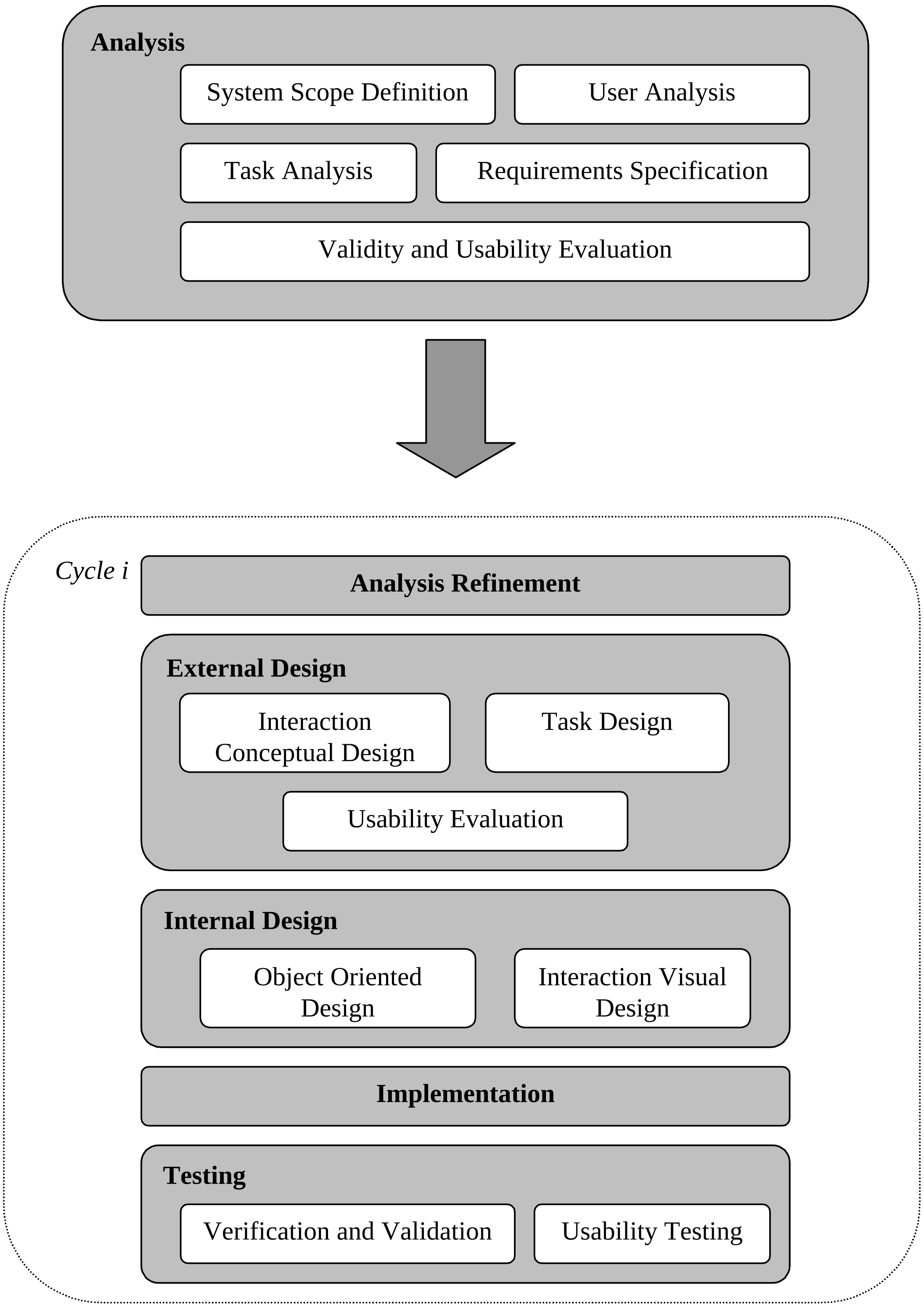


**Figure 1 Joint Development Process**

After the system is deployed, maintenance begins and it can be considered as additional cycles in the development. Maintenance cycles are driven by customer or user requests and some of the activities showed in figure 1 may be lighter than in previous development cycles.

Table 1 classifies the activities of the joint development process in the ones belonging to object oriented development and the ones from HCI. Please note that some activities combine both sources, such as *Requirements Specification*.

| ***Joint Development Process*** | **Object Oriented Development** | **HCI** |
|---|---|---|
| *Analysis* | System Scope Definition | |
| | | User Analysis |
| | Task Analysis | |
| | Validity and Usability Evaluation | |
| | Requirements Specification | |
| *External Design* | | Interaction Conceptual Design |
| | Task Design | |
| | | Usability Evaluation |
| *Internal Design* | Object Oriented Design | |
| | | Interaction Visual Design |
| *Implementation* | Implementation | |
| *Testing* | Verification and Validation | Usability Testing |

**Table 1 Activities of the joint development process classified according to their belonging to either object oriented development or HCI**

## 4 Usability Techniques in the Joint Development Process

HCI offers numerous techniques to be used for different project characteristics and for different usability purposes. We have chosen the ones more valuable for a broad variety of systems, considering specially their applicability from a software engineer point of view. Most of them can be applied with moderate usability training.

In the following sections we present the usability techniques we recommend. They are structured according to the phase of the joint development process where they can be applied. Table 2 summarizes this information.

### 4.1 Analysis

The activity of *System Scope Definition* will be highly dependent on the kind of system to be built. For traditional systems it can be just a short description of what the system is intended to do, but for innovative systems the set of techniques known as Holistic Design [10] can yield an adequate definition of what the system will do and how it will be like. This kind of definition can be called "Product Vision" [5]. For systems built from scratch, but not necessarily so innovative, Needs Analysis [2] would suffice.

*User Analysis* can be performed for in-site developments or tailored systems by means of site visits. A variant of sites visits is the Contextual Design approach [11].

The activities mentioned above feed the *Task Analysis* activity. The identified tasks are modeled following the technique of use cases, but with the focus of tasks [7] as mentioned earlier.

*Requirements Specification* is a traditional Software Engineering activity, but it will be performed in parallel with Task Analysis. Both activities are complementary, because identifying tasks can help to discover new requirements, and requirements must be related to tasks. Furthermore, we will include a set of operationally-defined Usability Specifications [2], so they can be checked by means of Usability Testing.

*Validity and Usability Evaluation* is a combination of Validity Evaluation performed on the requirements and usability evaluation. The latter can be performed using paper prototypes (sometimes called paper mock-ups [4]).

## 4.2 Analysis Refinement

At the beginning of a development cycle the analysis documents are revised. Therefore, all the activities described above for Analysis can be applied again in this phase.

## 4.3 External Design

Design has been divided into two phases: External and Internal Design. We want to deal first with the part of the system the user is aware of (External Design), that is the way of working with the system plus the elements of the graphical user interface. The former is addressed in Task Design and the latter in the Interaction Conceptual Design.

In the activity of *Task Design* every task (or use case) identified in Task Analysis is  specified in detail, in order to design precisely the interaction between the user and the system. Techniques described above for *Task Analysis* can apply. The structure for the detailed tasks produced can be the one proposed by Larman [9] for use cases in expanded format. The techniques mentioned for Task Analysis can be applied here as well, specifically where they address the design of the tasks the new system will offer.

We model the user interface elements in the *Interaction Conceptual Design*. It is a genuine creative activity, for which we cannot find a process whose application guarantees usable designs. Nevertheless, usability experts have gathered valuable Design Guidelines (like the ones in [3],  [4] and [12]), which are the basic guidance for newcomers to the field. We consider that the existing interaction notations are too formal to be applied by average software developers.

We can evaluate the result of both activities through *Usability Evaluation*. It can be performed either by Heuristic Evaluation[4], by Collaborative Usability Inspection[3], or by informal usability testing with users, but always keeping a focus on Formative Evaluation[2]. It is not that much checking measurable levels against previously defined desired values, as it is to get from the evaluation ideas of how to improve the usability of the interaction design.

## 4.4 Internal Design

This phase is mostly devoted to traditional Software Engineering activities, being *Visual Interaction Design* the only HCI activity. Design tips for this activity can be found in [12] and [13].

## 4.5 Construction

No specific usability techniques can be applied in this phase.

## 4.6 Testing

Along with the Software Engineering activities of *Verification and Validation, Usability Testing* is performed at a laboratory with real users [14]. With the results the developing team will be able to assess the fulfillment of the usability specifications defined in the *Requirements Specification*. When Acceptance Tests are performed, they can include usability testing of specific usability criteria [12].

### 4.6.1 Maintenance Cycles

After system deployment, maintenance can be performed  with a usability perspective as well.

Usability evaluation can be enriched with User-Performance Data Logging [4][12], User Satisfaction Questionnaires [12], and other kinds of user feedback such as beta testing or trouble reporting[12].

When users are organized in Focus Groups [4][12], they can provide feedback information more valuable than individual interviews, as Focus Groups are more representative of the user population.

| ***Joint Development Process*** | **Usability Techniques** | **Bibliographical Source** |
|---|---|---|

| | | |
|---|---|---|
| *Analysis* | Holistic Design | [10] |
| | Product Vision | [5] |
| | Needs Analysis | [2] |
| | Contextual Design | [11] |
| | Task Analysis | [7] |
| | Paper Prototypes | [4] |
| | Usability Specifications | [2] |
| *External Design* | Design Guidelines | [3] [4] [12] |
| | Heuristic Evaluation | [4] |
| | Collaborative Usability Inspection | [3] |
| | Formative Evaluation | [2] |
| *Internal Design* | Design tips for Visual Interaction Design | [12] [13] |
| *Testing* | Usability Testing | [14] |
| | User-Performance Data Logging | [4][12] |
| | User Satisfaction Questionnaires | [12] |
| | Trouble Reporting | [12] |
| | Focus Groups | [4][12] |

**Table 2 Usability Techniques to be applied in each Phase of the Joint Development Process**

## 5 Conclusions and Future Directions

The need for integration of usability techniques into an object oriented development process has been discussed. HCI does not offer a satisfactory process in Software Engineering terms. Additionally, current Software Engineering processes don't address properly usability issues, therefore producing unusable software products. Some proposals are beginning to emerge [5][6], but they are not generally applicable studies, instead they address specific cases in particular development organizations. A generic process where usability activities are integrated has been proposed, including the specific usability techniques to be applied in each phase according to the characteristics of the project.

Our experience has shown us that developers with a Software Engineering background regard usability activities as a nuisance, specially when company-wide rules require passing some kind of usability evaluation. In a development environment where development time is critical, and when reducing the time-to-market is the main objective, usability activities are perceived as a considerable delay in the project schedule. We have proposed in the joint development process a parallelization of HCI activities with Software Engineering ones where possible, in order to reduce at a minimum the increase in development time.

Software engineers require a high degree of flexibility when adapting to the joint process, as the HCI philosophy can be hard to fit together with a traditional Software Engineering background. Organizational and cultural change needs to be managed carefully.

Some activities in the joint development process combine Software Engineering activities with HCI ones. This combined techniques will have to be defined at a more detailed level, especially the ones belonging to the Analysis phase.

Finally, the application of the process to a variety of projects is needed to define a detailed guide of how to tailor the joint development process to specific projects and software development organizations. The relationship with management activities can be detailed as well after application to industry projects.

## References

[1] X. Ferré, N. Juristo, H. Windl, L. Constantine. *Usability Basics for Software Developers*. IEEE Software, vol.18, no.1, January/February 2001. pp. 22-29.

[2] D. Hix, H.R. Hartson. *Developing User Interfaces: Ensuring Usability Through Product and Process*. John Wiley and Sons, 1993.

[3] L. L. Constantine, L. A. D. Lockwood. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Addison-Wesley, 1999.

[4] J. Nielsen. *Usability Engineering*. AP Professional,  1993.

[5] J. Anderson, F.Fleek, K. Garrity, F. Drake. *Integrating Usability Techniques into Software Development*. IEEE Software, vol.18, no.1. January/February 2001. pp. 46-53.

[6] K. Radle, S. Young. *Partnering Usability with Development: How Three Organizations Succeeded*. IEEE Software, vol.18, no.1, January/February 2001. pp. 38-45.

[8] I. Jacobson, G. Booch, J. Rumbaugh. *The Unified Software Development Process*. Addison Wesley, 1999.

[7] J.T. Hackos, J.C. Redish.*User and Task Analysis for Interface Design*. John Wiley & Sons, 1998.

[9] C. Larman. *Applying UML and Patterns*. Prentice Hall, New Jersey, 1998.

[10] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, T. Carey. *Human-Computer Interaction*. Addison Wesley, 1994.

[11] H. Beyer, K. Holtzblatt. *Contextual Design: A Customer-Centered Approach to Systems Design.* Morgan Kaufmann Publishers, 1997.

[12] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, Reading, MA, 1998.

[13] K. Mullet, D. Sano. *Designing Visual Interfaces: Communication Oriented Techniques.* Prentice Hall, 1994.

[14] J. S. Dumas, J. C. Redish. *A Practical Guide to Usability Testing*. Intellect, 1999.

# Modeling the Usability Evaluation Process with the Perspective of Developing a Computer Aided Usability Evaluation *(CAUE)* System

**Michael Gellner, Peter Forbrig**
Software Engineering Group,
Department of Computer Science,
University of Rostock
Albert-Einstein-Str. 21
Rostock 18051Germany
++49 (381) 498-34 33
[mgellner|pforbrig] @informatik.uni-rostock.de

**ABSTRACT**
Usability testing deals with improving the usability of the inspected artefacts. The process that leads to these results is hardly known by people who are not used to work in that field. Only large companies are able to engage usability engineers and to install usability laboratories. So many systems to be used by end users are build by the vast majority of designers and developers that are not able to conduct usability evaluations.

It is considered as a fact that usability results in usability efforts which have to be accompanied by usability evaluation. As a consequence the usability of systems can be improved by increasing the efficiency of the usability process. Today the usability inspection works with only small support through computer-systems whereas the complete process could be supported. To fill this leak in future we try to understand the usability evaluation as a process comparable to the process of software engineering.

**Keywords**
Guidelines, usability engineering, usability testing, usability evaluation

## USABILITY EVALUATION AND THE USABILITY OF INTERACTIVE SYSTEMS

Good usability in common from consumer goods and especially from software is desired and expected today. Users do not tolerate furthermore the fact that a system is hard to use. More and more there is a competition which leads to situation in that such products cannot be sold to customers. What features a product had had does not interest if they were not to find or to handle because their representation could not be interpreted [5]. One more important development is that since the 1st January 2000 the whole extant of the European Screens Directive 90/270/EEC becomes valid in Germany, employees can accuse employers, if workspaces use software that does not follow ergonomic rules [7], since checking the existing software at usability belongs now to the employers duties[1]. The caused pressure employers will effect to the software producers in that matter, since such verdicts will lead to other demands if new software will be bought and to other interpretations of existing service agreements.

Adequate usability of products is a result that depends on efforts, similar to results in other domains. The development of acceptable user-interfaces requires specific knowledge and tools on the one hand and a resource intensive engagement in usability testing on the other hand. Peculiar by smaller projects the usability efforts can be about half of the whole budget. So usability testing and evaluation are omitted by such projects, often with the justification, that the effort for the user is always small in a small system. Consequently with the complexity of a system there is more that can be done with it. The user can compose more steps and the usability effort increase in the same degree. But there is a point[2], from that on, such tests are not more to avoid. Inconsistencies of subsystems to each other can for example only appear, if there are such systems. An easy to use complex system is hardly to create due to rules and styleguides, for that reason usability studies actually take place.

Usability evaluation often happens in the last phase of the development cycle. That is the case in smaller environments, where ergonomic laymen practice those evaluations. There usability testing is considered as a part of the testing-phase. Functional and logical errors will be fixed certainly in that situation, whereas a user-interface that misses the target will not be changed. Normally mistakes

[1] In some European countries the directive is valid, in the other ones it will become valid, since every partipiciant of the European Community has to turn this directive to national law.

[2] that depends from a project (complexity, size, programming interfaces, amount of components etc.)

that are found as late as this are expensive to fix, often there are conceptional problems, which become discovered. Methods to analyze the usability of a system in earlier states, are often unknown in such smaller environments.

## LONG TERM OBJECTIVE COMPUTER AIDED USABILITY EVALUATION *(CAUE)*

### Motivation

The process, that wants to help systems to become ergonomically, is obviously neither practical nor easy to learn, in fact it is not ergonomic [15]. There is no doubt in this field that this can cause losses on resources and efficiency. Since the process and its steps will not alter in the future tools for a better handling were helpful and could increase the transparency of the whole process. Perspectively this work and follow-up works should lead to a possibility of a round-trip usability evaluation. The title for this kind of systems is *Computer Aided Usability Evaluation-Tools (CAUE-Tools).* That is related to the *Computer Aided Software Engineerings (CASE),* that realises for the software engineering, what CAUE will aim for the field of usability evaluation. In the following the existing approaches and views on the usability evaluation process are shown. Based on this a new model will be presented, since the now existing views are not optimal for our purpose.

### Approaches

#### *Software centered evaluation vs.user centered design*

The evaluation of software is from the perspective of the usability based on two approaches, software oriented or user centred.

- Firstly a system is compared with all valid rules, that means to the common rules of human computer interfaces, the platform specific standards and possibly to rules of its working context. Divergences to the rules lead to a decreasing value for the system. Representative for this a way of evaluating the usability of software are inspection methods like EVADIS, GOMS or K-L-M.
- The user-centred design looks not so much on the question if a system sticks to the rules but on the relation from users to the system. As experience shows even software that confirms with all rules can lead to usability problems. For example a solution that matches the mental model of the user can be superior to another one that does not reach such a mental projection but follows all the rules.

#### *EVADIS*

The self set target of EVADIS developers is to help an usability engineer by checking a software system objectively [18]. With EVADIS the following steps have to be done:

- Finding test questions
- Asking for the properties of the users
- Collecting the test questions
- Valuing the questions
- Creating a test report

Despite the fact that the usability expert only compares behavior and design of a system with lots of questions, this method lays claim to register the ergonomic properties of a system holisticall [9].

In fact with EVADIS it is possible to decide weather a system follows a standard or not, if it is used despite the organizational overhead it brings with [3]. Limitations in the usefulness of EVADIS are:

- Following a standard is not automatically similar to be usable, furthermore norm full-filling is not implicit practical, EVADIS is based on nothing but standards. For example without a testing person the subjective contentment of a user cannot be considered.
- In a comparison of two solutions where incidentally the first one matches the users mental model but without following the known usability rules and another one misses this target but does not break any of these rules only the second one will be followed up if advises calculated by EVADIS will become foundation of the following implementations.
- New concepts require a lot of maintenance or their usage results automatically in norm violations. So this subtle web of questions needs permanently to be renewed. Small alternations - for example the implementation of dockable toolbars – are often used for a long time commonly until they are ‚justified‘ through a new styleguide.
- Learnability cannot be explored. It is nearly impossible to estimate what efforts are necessary to learn to work with a system. Since EVADIS works without viewing testing persons this important mark is as a fact ignored for really much applications.
- The EVADIS process with hundreds of detailed questions needs a product to judge. Because of the very detailed questions a nearly ready product has to be the subject of a test with EVADIS. It is not meaningful to do such a fine work with a prototype in that positions of elements as well as numbers and even kinds of graphical elements will still alter.

From the point of view of a usability tester that not only wants to judge the usability but rather tries to improve that property of a system such a testing method is useless. EVADIS follows a product cycle but does not integrate itself in that process for example as a kind of a subcycle. Since changes in an advanced product become more expensive around a factor of 10 with every taken phase, following the last phase causes horrible costs if there were necessary alternations (what is highly probable), for an improvement oriented usability testing EVADIS intervenes definitely to late [1], [2]. EVADIS maps the usability evaluation process in no way, it substitutes him with an own one, so properties for the conventional and methodical heterogeneous process cannot be derived from EVADIS.

*GOMS*

The term *GOMS* stands for the four components goals, operators, methods and selection rules, the foundations of GOMS. This method of task analysis divides a task recursively in subtasks until there are only atomic steps, which are the operators. Their effort can be measured and based on such operators a whole process can be estimated. Indeed it is possible to get an imagination of the kind and the time a task needs to be fulfilled[3] [16], [17]. Limitations are easy to reach, when the works that have to be analyzed are complex or with an irregular workflow tasks cannot be described with GOMS. Normally the majority of tasks is neither fulfilled by experts nor can they end always without any errors, but these are cases that GOMS is not optimal for and from a central meaning in the field of usability evaluation [8].

The results won by GOMS can be taken as a kind of references e.g. for analyzing learning curves but they are not proper for comparing them with different user levels. So GOMS is a method that is suited for analyzing a special class of works whereas it is not a tool for creating scenarios that shall be taken as scales in usability tests.

*the usability engineering process from the perspective of the user centered design (UCD)*

**View of the process from J. Nielsen**

A first source for the usability engineering process following the approach of the user centered design (UCD) can be found in [8], see *Table 1.*

*Table 1 Process of Usability Evaluation due to Nielsen [8].*

1. Know The user
   a. Individual user characteristics
   b. The user's current and desired tasks
   c. Functional analysis
   d. The evolution of the user and the job
2. Competitive analysis
3. Setting usability goals
   a. Financial impact analysis
4. Parallel design
5. Participatory design
6. Coordinated design of the total interface
7. Apply guidelines and heuristic analysis
8. Prototyping
9. Empirical testing
10. Iterative Design
    a. Capture design rationale
11. Collect feedback from field use

Although this representation seems to be complete it is for the following reasons not proper as a starting point for the further modeling of a CAUE system.

For our purpose some items (e.g. item 4 and 8) are hard to realize, since Nielsen presumes here possibilities of the usability engineer that he has when production and usability testing happens inhouse – a situation that cannot be assumed everywhere. If usability testing executed by external engineers should be considered also such a presumption is a kind of invalid limitation.

Often an external usability laboratory is invoked, since internal recommendations for the development could not be established [13] and sometimes evaluating activities are required to be done externally (founded by the quality management). From our point of view a mixture of aspects from software development cycles with that of the usability evaluation cycle is not practical, since this complicates the analysis of the single phases.

Similar to Nielsens model are the ones in [10] and [12] and analog to that they restrict to a process that works only inside a company[4]. Because of that facts it seems more appropriate to consider the mentioned cycles as production cycles (or as parts of such ones), since they leave the actual usability work[5] in many points.

***Process description from J. Rubin***

Rubin showed in 1994 [11] one more approach, that watches the actions of the usability tester:

1. Developing the Test Plan
2. Selecting and Acquiring Participants
3. Preparing the Test Materials
4. Conducting the Test
5. Debriefing the Participant
6. Transforming Data Into Findings and Recommendations

*Critic on Rubins Model*

---

[3] So it can be modeled how in a word processor a file has to be opened. Then as atomic considered could be the mouse click on the *'file'* menu, the mouse click on the menu point 'open' and the following actions inside the dialog.

[4] Indeed all the three authors wrote their articles from the perspective of usability testers *in* companies (Nielsen comes from SUN- Microsystems, Pflüger from the Schweizerische Kreditanstalt and Stimart from GE Information Systems) where the own produced software has to be analyzed.

[5] that exact description it is looked for

Although Rubins model describes only phases that are indeed part of the usability process, it does not suit as the theoretical foundation for a CAUE-System. On the one hand important aspects are missing and on the other hand some representations are not optimal way for later works:

- The enormous time-consuming phase of the data preparation in that the whole amount of the rough material is reduced to data that can be evaluated is missing in Rubins model.
- The evaluation and the creation (that means the *writing*) of the study are not distinguished in any way. Since CAUE will support the standards for quality management[6] that is a strong limitation. If a company is obliged to document the complete production process a usability study is an important document. Writing a study for that purpose is a work that binds an usability tester up to a week per project/test series. Because of this effort the writing should be shared from the evaluating phase. And from a methodical point of view it is unclean to differ not between the central phase of evaluating and the only formal motivated writing. In a CAUE system obviously absolute different modules could support these actions, representations that neglecting this fact will lead to further consequences with a high degree of probability.

Indeed Rubins view is near on the practice, similar to the intuitive work and easy to set. For the CAUE approach this real features for practical work miss important formal aspects.

*Analysis of the usability evaluation process*

A *CAUE* system should lead through the whole usability evaluation process and accompany through every single step as detailed as possible. To aim this goal it has to be analyzed from what components this process does consist of and which parts of it can be supported.

A first simple view on the process offers the following schema: *working materials* and *templates* from the usability tester lead in collaboration with the producers/developers and with testing persons to a certain product, to the usability study (see figure 1).

In the first step of this three phase model *input* means the activities from the usability testers (ut) to produce working materials like questionnaires or paper and pencil pattern. In the second step input stands for the testing persons (tp) activities. In this phase the developed working materials are transferred to a kind of (pre) results[7]. To the end the last input means again the usability testers activities. At this points it stands for the work with the data that leads to the final recommendations i.e. the actually evaluation, the recognition from relevant facts out of the time series and prepared materials.

In a next approach the first view that can be seen as an kind of an outside view becomes more refined (see figure 2): An amount of parameters leads to specific testing methods to be chosen from a set of available methods, so that the global goals of the usability evaluation process will be reached. The testing methods determine widely the engaged materials. Today the literature proposes nearly 60 methods for conducting usability studies. Normally to every method that is used usually, fitting strategies and materials have to be prepared. Every available information about the users and the purpose of the system should enter in the preparations. Mostly the principal states which aspects (common acceptance, suitability for the task, self descriptiveness) have to be analyzed and with which efforts. The term *usability test* normally demands that tests with testing persons take place (see *Software centered evaluation vs.user centered design*). In such investigations the participants turn as before the testing materials to working materials, e.g. questionnaires turn to questionnaires that can be evaluated, there are voluminous audio and video sequences and depending to the methods further documents or media, these are intermediary results. From those the final results can be detected by quantitative and qualitative analysis.

After this evaluation the final results needs a representation, in most of the cases this is a document that describes the whole work, the used methods to test and evaluate as well as the results and recommendations or the corrections (see figure 2).[8]

Although this view again already mediates a complex impression, further considerations of the process are necessary since with that model view only trivial requirements can be fulfilled and efforts cannot be estimated. Whereas today there is a widely knowledge about the principles that

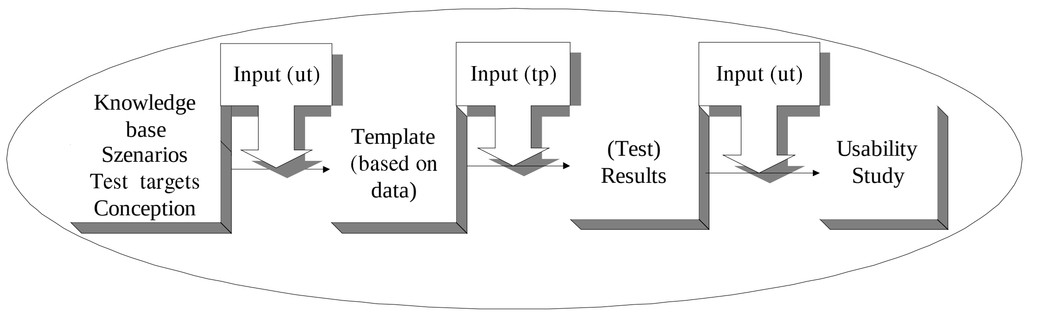


*Figure 1 a view on the usability engineering process*

d of result in this model,
ıe, it works the same way
interview strategy.

› the time a project is ac-
information exchange. If
ɔtotype to be tested, the
:ead of this the document
ing documentation if us-
developer work together
esults intensively.

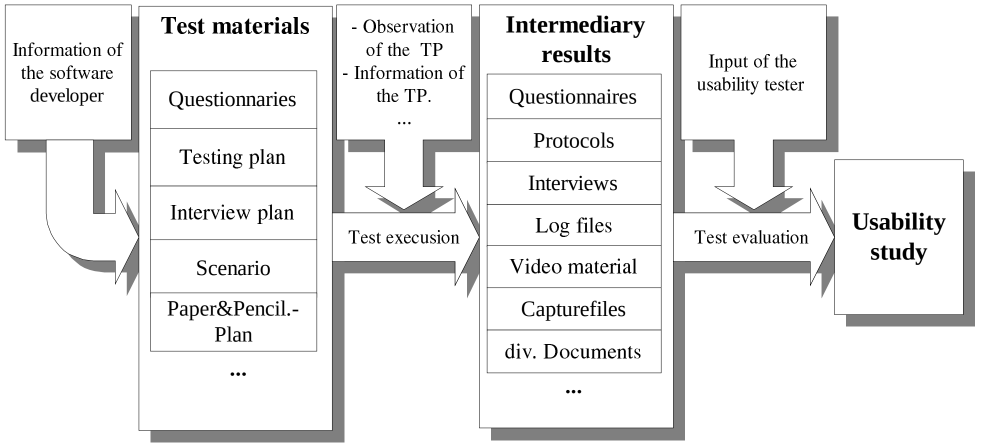


*Figure 2 Refined view on the usability evaluation process*

have to be followed from an ergonomic perspective[9], the process that measures the success of that steps is hardly analyzed. Sources are rare, especially from the sight of creating supporting computer based systems.

The paper from [14] and [4] that occupy with the software support of the usability evaluation process give a description of that a miss also and set without a concrete view on the ISO 9241 standard or on the EVADIS system.

Since usability testing is concerning to the tendency to user centered design (UCD) today an activity that works with (test) users [3], [11] or similar representatives experimental proceedings are commonly used. In contradiction to schematic or formal methods possibilities to feedback about more layers are necessary.

**The View from a CAUE System on the Usability Evaluation Process – Eight Phases with Feedback Paths**

The following in figure 3 shown phase oriented model fulfills the demands that were made implicitly until here:

- The activities of usability testing are gathered.
- The model restricts on the relevant aspects (see Critic on Nielsens Model)
- The described steps in the model fit to significantly distinguishable steps[10]
- It offers a practicable and handy basis for modeling and designing a *CAUE* system

In addition to that the steps concerning this model can be implemented separately so that the single products can be used apart from the other ones. The model can be connected to the known models for the software development part (see figure 3, 4 and 5) and follows self the waterfall metaphor. Testing from systems is so from ergonomic perspective a process similar to known processes in the producing sector of the industry or in software engineering and attended from methods of diverse social sciences and the psychology. The following model consists of eight phases which are described in figure 3. Different to the waterfall model in software engineering [2] the feedback paths must not restrict one layer, every feedback path is possible except the deepest layer only.

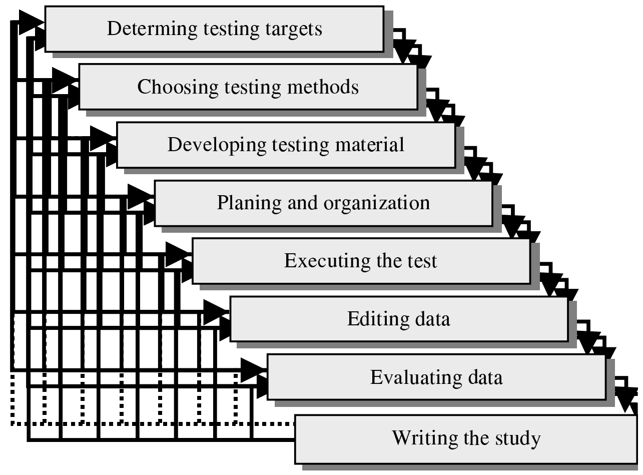


*Figure 3 Eight phases model for usability testing processes and the possible feedback paths*

Ultimately in every phase it can put out e.g. that methods do not suit or that important parts of the systems can not be reached during the execution of the test. With such data no reasonable work no reasonable work is possible, only a further execution after a different planing and conception is here recommendable, that requires jumping back as much layers as necessary. Only after having written the study no feedback seems to be helpful. If the process reached that point all work before ended successful, otherwise that state would not have been reached. Indeed further testing can take place, but formal this is considered as a new study and that can deal with a new version of a former known system. The single phases and their meaning:

*Determine testing targets:* As usual by the planning from projects the exact subject of it has to be declared and sometimes even to be found.

[9] And what mistakes should be avoided

[10] Compare e.g. with Nielsen ‚Prototyping‘, that describes nor when this step has to be done into the usability engineering cycle neither when it is done. Even the target is not directly obvious, since a method like that needs a software developer, a requirement that is not really necessary for working in the field of usability testing.

*Choosing testing methods:* On the whole the methods depend from the declared targets, the resources and the phase of the software project.

*Developing testing material:* The testing material is determined through the chosen testing methods.[11]

*Planing and organization:* To that point all indirect efforts for conducting usability tests are counted, that means efforts for corresponding with participants, room and personal planning, installations, technical devices and so on.

*Execution of the tests:* The phase in that the chosen methods are applied together with the testing persons. Adequate media are used to record the sessions. Although the execution and its correctness have a central meaning, this phase does not cause the highest efforts.

*Editing data:* The testing data does not come in a state, that allows evaluating, hours of video sequences need to be filtered, answers in questionnaires need to be counted trivially. Although this phase of the contents can be considered as a prephase of the following evaluation, the separate modeling offers advantages:

I. The editing phase is the one that causes the highest time efforts in the whole usability engineering cycle (motivating aspect).

II. This phase offers enormous possibilities for supports through computer based systems to minimize the disproportionate efforts.

III. To assign that phase to the following evaluation phase would be improper. Both the efforts and the multiple solution approaches justify that consideration, since this seams nearer to a reasonable implementation.

*Evaluating data:* That means the quantitative and qualitative analysis of the edited data.

*Writing the study:* Depending to the requirements of the principal the results are written down. Intensive working together can cause that every relevant information is transferred between developer and tester. In this the study has only the meaning of a business report. If the principal has to prove that he used external consulting in view of quality management the study can become a central meaning, more than any consulting that happened indeed. A study contains documentary aspects, relevant results or a detailed list of every remarkable point and further a description of the progression for the corrections. An more internal view in the single phases will be subject to a follow up paper.

**Integrating the eight phase model into the software lifecycle**

Rubin describes in [11], that in essence the current testing methods belong to one of the following four testing classes:

- *Exploratory test:* Different alternative solutions for a system should compared. The focus is the width of a concept, the depth of the prototypes is neglected.
- *Assessment Test:* This kinds of tests require more concrete systems, since the practicebility from dialogs is checked with real testing persons. The investigations analyze the usable depth.
- *Validation Test:* The analyzed system is treated as a ready to run product. Focus is ion reasons for bad system behavior.
- *Comparison Test*: Focus is the consideration of different alternatives thus comparison tests base on the three other testing classes.

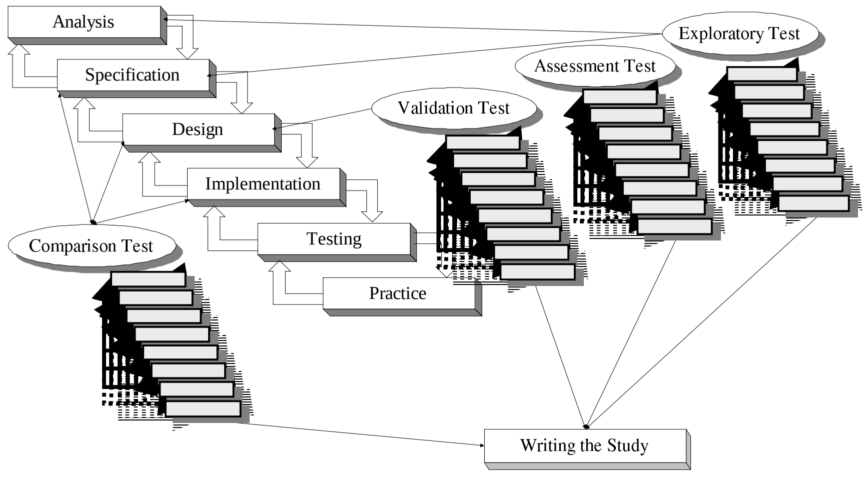


*Figure 4 Connection of the eight phase model for usability testing with the waterfall model for software development following the four testing strategies due to [10] with uncoupling the last step.*

What kind of test due to Rubin ultimately takes place, is determined from the phase of the project, from the number of drafts and from the targets of the testing. All testing classes can be subject of the shown eight phase model that again can be used in every phase of the software lifecycle as a sub cycle, figure 4 and figure 5 show this for two different software lifecycle models.

[11] And as the case may be to the extant of the resources.

The first seven phases of the eight phase model are taken in all testing series. They depend from the amount of problems if the model can passed linearly or if feedback ways are necessary (see paragraph *The View from a CAUE System on the Usability Evaluation Process – Eight Phases with Feedback Paths*). The multiple repetition of the model is the normal case if a product is accompanied through a longer period whereas the single run through a nearly ready product methodically causes problems. Serious changes would come along with high efforts and costs, so it is the rule that they remain undone.

From there a early integration of usability aspects into the development cycle is wanted. So a testing cycle (a run through the first seven phases of the eight phase model) from a certain state on after every extensive modification of the user interface can be helpful. Writing a concluding study after such steps is not necessary[12], in figure 4 and figure 5 this step is suggested separately on the very end of the investigation.

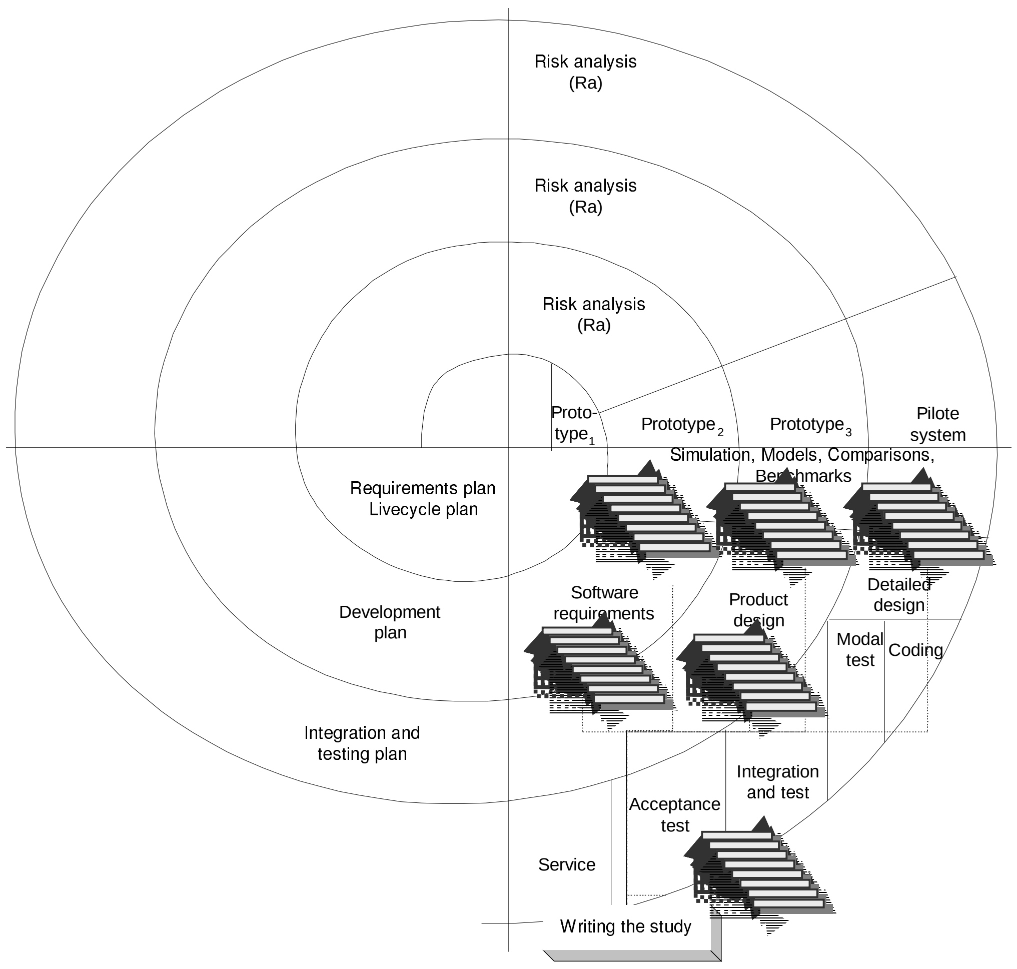


*Figure 5 connection of the eight phase model with the spiral model*

## CONCLUSION

The shown eight phase model of usability evaluation fulfills the present requirements to describe the process comprehensive, detailed and in a kind that facilitates the further development. With the eight phase model the further way can be seen clearly: the single phases need to be worked out. Then different interfaces to the software engineering process should be analyzed since a brisk information flow between usability tester and software developer are wanted. At this point a high potential of optimizing can be seen. A milestone is reached when the usability evaluation process can be controlled only by an experienced user without knowledge in software ergonomics or in software technology, that is one of the declared targets of CAUE. With such a state a further development was possible. A system that follows this principles could increase the efficiency of usability tests enormously. Another field that can be worked on with such a system is the analysis of the fundamental testing methods, especially in combination with knowledge bases that could come with a CAUE system. Also different ways of integrating CAUE systems in CASE tools and in development systems directly should be found, since that offers the shortest distances between the counterpart twins development and usability.

[12] Except the situation that the study serves the purpose of documenting the efforts in a quality management frame.

## REFERENCES

1. Bevan, N., *Integrating Usability into the Development Lifecycle.* In: 1st International Conference on Applied Ergonomics (ICAE’96), Istanbul, June 1996.
2. Balzert, H., *Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Spektrum, Akad. Verl., Heidelberg, 1998.

3. Denning, P. and Dargan, P., *Action-Centered Design*. In: Terry Winograd (Editor) *Bringing Design to Software*. Addison Wesley, 1996.

4. Hartwig, R., *Gestaltung und Bewertung der Gebrauchstauglichkeit von Software, Überprüfung und Machbarkeit am Beispiel der Benutzungsoberfläche eines Moduls eines Schichtplangestaltungssystems.* Diplomarbeit, Universität Oldenburg, 1997.

5. Hayne, C., *Software Engineering for Usability. Integration of human factors for user interfaces into the software development life cycle.* Untersuchung für General DataComm's Multimedia R&D Centre im Rahmen des *Telecommunications Multimedia Program*, Montreal 1996.

6. Hampe-Neteler, W., Baggen, R., Zurheiden, C., *Die Ergonomen kommen. Was die Bildschirmarbeitsverordnung für Software bedeutet*. c't, Nr. 25/99: S. 100 - 102.

7. Mayhew, D. J., The Usability Engineering Lifecycle. Morgan Kaufmann Publishers, Inc., Kalifornien, San Francisco, 1999.

8. Nielsen, J., *Usability Engineering*. AP Proffessional, New Jersey 1993.

9. Oppermann, R., u. a., *Software-ergonomische Evaluation. Der Leitfaden EVADIS II.*, 2., new edit., de Gruyter, Berlin, 1992.

10. Pflüger, J. P., *Usability Engineering: Organisation der Kreativität im EDV-Projektmanagement, Software-Ergonomie in der Praxis.* Projecta Verlag, Winterthur, 1992.

11. Rubin, J., *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*, John Wiley & Sons, Inc., 1994.

12. Smit, E., *Usable usability evaluation: If the montain won't come to Mohammed, Mohammed must go to the mountain.* In Patrick W. Jordan (Editor), *Usability Evaluation in Industry*. Taylor & Francis, London, 1996.

13. Stimart, R. P., *GE Information Services.* In Michael E. Wiklund (Editor), *Usability in Practice, How Companies Develop User-Friendly Products.* AP Professional, Massachusetts, 1994.

14. Strapetz, W. W., *Implementierung des ISO 9241-Evaluators. Vergleich der Dialogentwicklung mittels der Methode des CASE-Tools ADW und der Methode Dialognetze/Constraints.* Diplomarbeit, Universität Wien, 1998.

15. Thovtrup, H., Nielsen, J., *Assessing the Usability of a User Interface Standard*, Proceedings ACM CHI'91 Conference, S. 335-341, New Orleans, 1991.

16. Wandmacher, J., *Ein Werkzeug für GOMS-Analysen zur Simulation und Bewertung von Prototypen beim Entwurf.* In G. Szwillus (Editor), Tagungsband PB'97: Prototypen für Benutzungsschnittstellen (pp. 35-42). Paderborn: Universität - Gesamthochschule Paderborn, FB Mathematik/Informatik. (Erschienen als *Notizen zu Interaktiven Systemen*, Heft 19, November 1997, S. 35 - 42).

17. Wandmacher, J., *GOMS-Analysen mit dem GOMS-Editor GOMSED.* Unterlagen zu GOMSED am Institut für Psychologie, Abteilung Angewandte Kognitionspsychologie Technische Universität Darmstadt, 1998

18. Reiterer, H. und Oppermann, R., *Evaluation of user interfaces: EVADIS II – a comprehensive evaluation approach.* In: Behavior & Information Technology, 3, S. 137 – 148.

# A Usability Designer at Work

Bengt Göransson[1,2]

[1]Uppsala University, Department of Human-Computer Interaction, P.O. Box 337, SE-751 05 Uppsala, Sweden

[2]Enea Redina, Smedsgränd 9, SE-753 20 Uppsala, Sweden
Email: Bengt.Goransson@enea.se

ABSTRACT
In this position paper I introduce the idea of having a usability designer role in development projects to enhance a user-centred design approach. This role is responsible for the usability in all phases of development, integrating usability design into the development process. It is my experience that there is a need for this kind of role, advocating for usability in the organisation and in all projects.

## 1. INTRODUCTION AND PROBLEM BACKGROUND

There is no sole and exact definition of user-centred design (UCD). John Karat from IBM classifies UCD as: “For me, UCD is an iterative process whose goal is the development of usable systems, achieved through involvement of potential users of a system in system design.” (Karat, 1996). I regard user-centred design as a process based on an attitude. The *process* must be drawn upon the key principles for developing usable systems articulated by Gould et al. (1997): Early—and continual—focus on users; empirical measurement; iterative design; and integrated design—wherein all aspects of usability evolve together. Further on, it demands the “users” of the process (the developers) to have a user-centred *attitude* and act accordingly. I also believe that to be really effective, user-centred design must become the standard operating procedure for a developing organisation. Otherwise it will always be questioned and degraded. If user-centred design becomes *the way* to develop systems in an organisation there is no longer a need to speak out loudly about it or to question it. No matter if it is an in-house organisation or some other type of organisation, support from management is crucial to achieve this. Further on, there must be an understanding between the development organisation and the organisation buying the system to work according to a user-centred design philosophy. It is important to consider usability and user-centred design as a part of the development process, rather than something that is added on. I have through my practice and research encountered some major obstacles for introducing user-centred design:

- There are problems in understanding and recognising user-centred design. State of the art development processes do not honour usability and user-centred design, but organisations think that it can be added on without any cost.
- Lack of competence in usability and user-centred design. These topics are not sufficiently integrated in higher education and practices.
- Usability is often taken for granted and does not get any attention.
- If a client in the tender process (where a client orders a system from a developer organisation) does not specifically order a usable system, i.e. have usability requirements built into the requirements specification, the developer organisation is reluctant to spend any additional resources on making the system usable.

Starting to use a UCD process and, further on, to use it on a regular basis, is a great challenge. It surely has to be a step-wise adjustment to the “new” paradigm and for most organisations it will never be a total shift, rather the integration of some activities and methods to their present process and arsenal of methods. In my practice and research, I have focussed on the development process and tried to challenge some of these obstacles from that angle. *Usability design* and the *usability designer* development role are attempts to do this.

## 2. RESEARCH METHODS

I work both as a practitioner and researcher and have the opportunity to use existing methods, and partly developing new methods and processes, putting them into practice and using them as activities in the development process. In this way I get the chance to study, analyse and reflect upon the true value of the proposed course of actions or procedures. As a result of these studies I can further improve the methods and practices to formulate new theories and so on. This is an iterative process that has the potential to engage all parties involved in a system development project. The result of such a project is not only the developed system, but also knowledge and experiences about the process itself and the practice of it. My research approach falls into the category of action research.

Action research is a methodology that has the dual aims of action and research (Dick, 1993):

- *Action* to bring about change in some community or organisation or program.
- *Research* to increase understanding on the part of the researcher or the client, or both (and often some wider community).

The mix of action and research can be tuned to the level that is accurate for the researcher's aims. One important aspect is that the researcher takes part in the studied situation, for instance a project, not just as an observer but also as a fully participating project member. Action research as a research method makes it possible for the researcher to apply his/her theories in practice, in a realistic work situation, take action and make a change in that situation. Action research is not like controlled experimental research with fixed parameters in a laboratory setting. Instead, action research projects are conducted in real life projects at the practitioner's work place.

## 3. DEFINING USABILITY DESIGN

Usability design is my attempt to "put a face" on user-centred design and try to get organisations and projects to start to adopt parts of the philosophy behind user-centred design. The main rationale behind the concept is that clients (buyers of software development) want the design solution. They are not particularly interested in all the fancy methods and theories that Human-Computer Interaction (HCI) and usability people talk about. I see usability design as an unpretentious and lightweight UCD process that can work in practice. Usability design is to some extent the marriage between usability engineering, and user interface and interaction design. For those familiar with the subject, it is supposed to be the best from two worlds: Jakob Nielsen (usability engineering) and Alan Cooper (interaction design). Usability design is also an expression that Gould et al. (1997) used when describing their principles for UCD. In an attempt to define usability design I like to say that usability design is: *a user-centred design approach for developing usable interactive systems, combining usability engineering with interaction design, and emphasising extensive active user involvement throughout the iterative process*. Usability design is one way to focus more on the solution, the design, than, e.g. usability engineering does, but still have one foot in usability and HCI. Is, then, usability design just another term for usability engineering? No, it is inspired by usability engineering but is focused more on design and integrated user-centred design. It is my experience that, in general, developing organisations have difficulty in understanding the full benefit and potential of "pure" usability engineering. Usability engineering as defined by Preece et al. (1994, p. 722): "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.", focuses traditionally on metrics for measuring usability. Here we have too much analysing and evaluating, and not enough of more pragmatic design solutions. On the other hand, just user interface and interaction design is not enough, so usability design may be regarded as weaving them together.

The figure to the right is a result from my research and practice, and describes roughly the steps in the development process that are focused on usability design aspects. The principles for user-centred design by Gould (see the upper right corner of the figure) are there to set the ground for the usability design process. The process should be used iteratively and can be used together with other development processes. The process contains activities that can be carried out with various methods. The exact selection of methods is done in an earlier stage (when planning the UCD process). It is still a framework and is not fully developed or evaluated. It serves the dual purpose to be an understandable and communicative arte-fact, and a process that guides the user-centred design process.

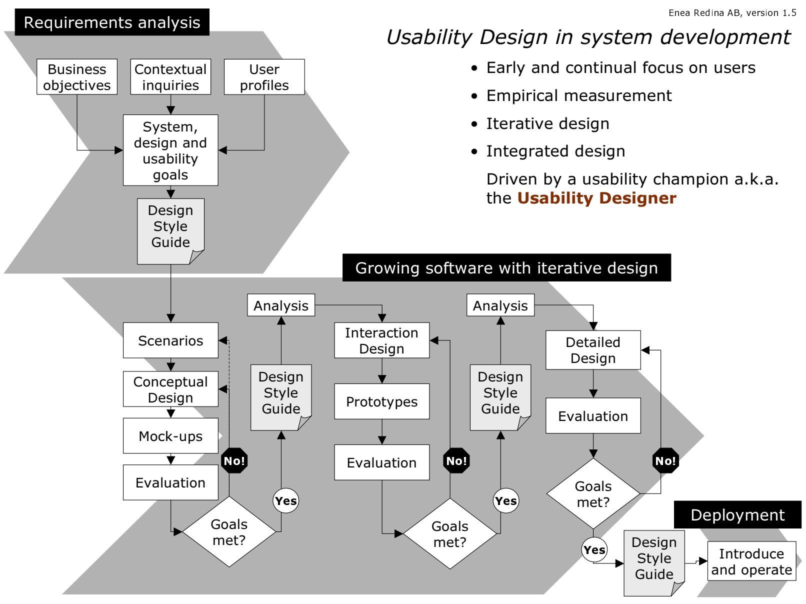


Figure 1: The usability design process.

## 4. USABILITY DESIGNER

The usability designer development role as a usability champion, and an advocate for usability and user-centred design is a representative for the usability design approach.

Background to the role was the urgent need for a practical way to really be able to practice usability and user-centred design. During my time in different industrial as well as in research projects, I had noticed the difficulties to bring in—and to practice—user-centred design. I had realised the obstacles for practicing user-centred design mentioned earlier. Many of the organisations I worked with were not committed to, or even aware of, usability. So I defined the usability designer role as a kind of usability champion on a user-centred design level rather than on a user interface design level, with the explicit purpose not to introduce just another user interface designer, but someone with the capability to work for users' best and user-centred design throughout the whole organisation and in all projects.

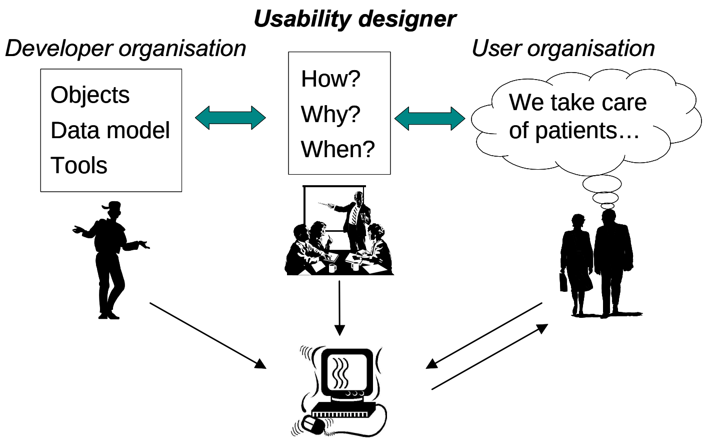


Figure 2: The usability designer as a development role to facilitate the user-centred design process.

The role is not an easy one, but when successful, promises to make large impact in organisations and projects. The characteristics of the role are:

- The usability designer is responsible for keeping the development process user-centred, focusing on usability aspects. Planning and performing activities related to the usability design process and making sure that the results of usability activities are further used in the development process, is very important for the usability designer. The role must also be given the authority to advocate for the users by management both in the development organisation and in the user organisation.
- It is crucial that the usability designer takes an active part in the design and development process, and does not only become another project manager. The usability designer can make a large impact by being present in most situations where the design of the system is discussed. By promoting a user's view in every situation, developers and others may always be forced to think twice before doing anything that would compromise usability.
- I emphasise the importance of a person participating in all the user-centred activities, to prevent valuable information from being lost in the transitions between the activities.
- The usability designer can to some extent be seen as a "discount" usability role, as it combines several skills in one role and in an efficient way copes with the user-centred design process.

## REFERENCES

Dick B., (1993), *You want to do an action research thesis?*, available on-line at http://www.scu.edu.au/schools/gcm/ar/art/arthesis.html, download date 2001-01-16.

Gould J. D., Boies S. J. & Ukelson J., (1997), *How to Design Usable Systems*, in Helander, Landauer & Prabhu (eds.), Handbook of Human-Computer Interaction, Elsevier Science B.V.

Karat J., (1996), *User Centred Design: Quality or Quackery?*, in the ACM/SIGCHI magazine, interactions july+august 1996.

Preece J., Rogers Y., Sharp H., Benyon D., Holland S. & Carey T., (1994), *Human-Computer Interaction*, Addison Wesley Publishing Company, Wokingham, England.

# USABILITY AS A TOOL FOR COMPETENCE DEVELOPMENT

*Stefan Holmlid*
*Human-centered Systems*
*IDA, Linköpings universitet*
*steho@ida.liu.se*
*(currently at LinLab, Ericsson Research*
*+46 13 28 4217)*

# Usability as a tool for competence development

The main focus for HCI has been the interactive artefacts, adapting them to their users. The potential of users adapting to technology is largely unexplored, despite the fact that companies spend more than ever on training their end-users. The research described here focus on the practices of the learning facilitators. Through an intervention project they assess the usefulness of a model of use quality for design and evaluation purposes.

## Adapting users

For many years the primary focus of the field of HCI has been the development of the interactive artefacts. The assumption has been that technology should be adapted to humans. A large amount of interesting and useful results has been achieved. As infrastructures and technology continues to develop we will experience even more progress.

At the same time users try to keep up. Companies spend more than ever on their employees competence development, and as part of that learning how to use the software. There is a large potential in users adapting to the technology. When it comes to learning work within HCI has mainly been concerned with users learning to operate the interface and its controls, e.g. Rosson (1983).

Interesting work giving new perspectives has been performed under the influence of minimalism, learner-centred design, and various aspects of computer supported learning.

The research problem has a split nature. On the one hand there is need for research and systematic exploration of, e.g. individual differences, in combination with changes in usability, use and learning (see e.g. Thomas (1996)). On the other hand there is need for research on methodology as well as models for use quality. The latter focuses on learning facilitator practices, and the construction of models of use quality that can be used formatively by facilitators in designing learning environments.

Since 1995 I have conducted an intervention research project together with competence developers at a large Swedish bank. The research goal has been to develop practice based models of use quality, that can be used for purposes of designing better products. Not only better learning environments, but also, in a longer perspective, better interactive artefacts.

We are in the middle of finsihing this project up, a PhD thesis is due October 2001. Early analysis, and preliminary, in the longer perspective, can be found at http://www.ida.liu.se/~steho/publications/index.htm.

## References

Rosson, M., B. (1983). Patterns of experience in text editing. In Human Factors in Computing Systems, CHI '83 conference proceedings, pp 171-175, Elsevier, Amsterdam.

Thomas, R., C. (1996). Long term exploration and use of a text editor. PhD thesis, University of Western Australia.

# Learning from traditional architects

Lorraine Johnston

*School of Information Technology*
*Swinburne Computer Human Interaction Laboratory*
*Swinburne University of Technology*
*Hawthorn, Australia 3122*
*ljohnston@swin.edu.au*
*Fax: +61 3 9214 5501*

## Abstract

*This paper focuses on lessons that can be learned from looking at the history of traditional architecture. When architects stayed aloof from the common building activities during the Industrial Revolution, a low standard of building resulted, and the needs of the people were made subservient to the needs of industry. Can we apply lessons from traditional architecture to software?*

*This paper reflects on the role of usability patterns and suggests there are probably other lessons which can be learned, for example, in terms of affective interfaces.*

**Keywords:** Usability, architect, software process, history of architecture

## 1 Introduction

Heath (1991), a practising architect, published a humorous book about traditional architects and their profession, writing:

> *"We all know what a perfect building is like. It never cracks or leaks. It finishes, if it ages at all, gracefully. The layout is convenient for its occupants, and explains itself to the stranger. The rooms are neither too large or too small; their shape is just right for what is done in them. . . . , yet it does not cost a fortune to run. . . . Cleaning and maintenance are easy. The building is secure against robbery, and resists vandalism, or never attracts it. . . . "*

Heath's description clearly flags the fact that many of the concerns of traditional architecture parallel those in the software industry, i.e. reliability, ease of use, fitness for purpose, efficiency, maintainability, security, etc. While there are numerous anecdotes about the fallibility of architects, we still value their services when we wish to construct a complex building—or in many instances simply to alter a home. We expect the architect to produce a design which is, among other things, usable. For example, we would prefer that disabled access was part of the original design, and not added in a piecemeal fashion afterwards. Similarly, we would like a building oriented to optimize the natural warmth and light in winter, while minimizing the impact of the hot sun in summer. Such attributes cannot be built in later, but must be designed in from the beginning.

Are there lessons to be learned from the discipline of (traditional) architecture about producing a usable product?

## 2 Architecture and History

The activity of architects is documented as early as the third millennium B.C., but some traces of architectural practice predate this substantially. For example, there are extant architectural drawings from as far back as ancient Egypt and Mesopotamia (Kostof, 1985)—the word "architect" actually comes from the Greek word meaning "master builder".

History records the architect as being the original master-builder, responsible, for example, for the design and construction of medieval churches. By the seventeenth century, the architect was still dependent on the guilds and their craftsmen, who included masons, carpenters, carvers and cabinet-makers, but usually had a clerk of works to take charge of the actual construction on the site (Richards, 1974). During the eighteenth and nineteenth centuries, the role of the architect changed, becoming less personal. The builder now controlled the craftsmen, and the architect visited the building site less frequently, communicating his instructions mainly through complex sets of diagrams (*ibid*). The period saw the divorce of design from construction, and architects became caught up in the revival of many different architectural styles. When the industrial revolution brought the need for different types of buildings, such as factories, mills and warehouses, architects tended to have little interest in coping with the construction needs of the industrial age (Stevens, 1965).

> "*. . . while the architectural champions of Gothic and Classic were making the headlines, hard-headed practical and industrious men were quietly changing the face of Britain with their railway stations and their viaducts, their mills, factories and housing estates.*
>
> *The factories and mills . . . and the housing, squalid, insanitary, shoddily constructed and unspeakably dreary, cried out for the vision and aesthetic touch of the architect, but they were left to builders and their factory-owning clients whose interests lay not in social welfare or visual beauty but in industrial expansion. The urban slum was born.*"

It was a new player who took responsibility for the design and building of these new types of structures—the civil engineer. Gradually he developed a scientific approach to the new constructional forms and new building materials. Eventually, the process of constructing large building works would involve a team of specialists: the architect, the engineer, the quantity surveyor and the contractor (Stevens, 1965).

During the eighteenth and nineteenth centuries, architects were rather sidelined with respect to changes to building style and thus the changes to society. They followed classical design principles and left others to deal with the requirements resulting from the industrial revolution.

It was not until the early twentieth century that architects reemerged as key players in the building industry. They came to appreciate the new materials and techniques, but also realised they had a vital role to play in bringing order and *humanity* into an increasingly chaotic world (Richards, 1974). Today, the architect works on the project not only with the structural engineer, but also with electrical, heating, lighting and acoustic engineers, not to mention interior designers.

An architect still specialises in design, but even more possesses *"a capacity for co-ordination, compromise, and negotiation, the ability to balance competing demands and needs and to appreciate the points of view of other professionals with their own desires."* (Kostof, 1977). Numerous books exist to introduce architecture students to their duties in terms of dealing with the range of people and events necessary for the life of a successful project, e.g. Harrigan & Neel (1996).

Kostof, p.335, also notes: *"To the new role of team coordinator, the architect would bring also a developed social conscience and a mission of service to society. ... the new architect was to seek out society's needs, identify and propose solutions for them, bring together the necessary skills, and operate as a member of a team ... "*

### 2.1 An observation

Traditional architects designed and built edifices for all types of people. When they forgot about the real needs of users, and technologists took over during the industrial revolution, the result was low quality. It was only when architects understood and worked with the advances in technology that they were able to contribute in a socially-meaningful way.

Software started as a specialist area, but today is used in a generalist way. In earlier days, when interactivity of software was not an issue, it was reasonable to have the software development driven by technologists. However, today interactive software predominates, and high quality is clearly not achieved unless the needs of the user are taken into account. Both the human and the technological factors are important.

The person who is today charged with the architectural design of a large software product has a role similar to that of the structural engineer. That person is part of a team and interacts with people of other specialities to ensure that the product will meet such requirements as performance, reliability and portability. Usability is not always included as a specific requirement, as technical experts often believe they know what the user needs—just as the civil engineer of the industrial revolution knew what was needed to build factories and railways! Providing a usable product is more often than not seen as merely the technical issue of providing a (graphical) interface which has a suitable layout—which in no way guarantees the user's needs are met. Thus only some of the requirements for software tend to be taken into account by the current incarnation of software architect.

## 3 Designing for usability

The development of design theory can be traced from the early architects such as Vitruvius through industrial design and into systems design. An architect designing a building must synthesize a solution which can resolve numerous conflicting forces. Most lists of principles for good software design include a requirement for the design to exhibit uniformity and integration (see, for example Pressman (1997)). Brooks contended that "conceptual integrity" is *the* most important consideration in system design" (p.42 (Brooks Jr., 1995)). He further says that every part should reflect the "same philosophies and the same balancing of desiderata".

### 3.1 Usability principles

People working in Human Computer Interaction (HCI) often talk of usability principles, but do these principles have a parallel in traditional architecture?

Mahemoff & Johnston (1998) considered that existing HCI guidelines were heavily focussed on graphical user interfaces, and gave little support for developing more general types of interfaces. They therefore abstracted from the usability properties given in the most popular guidelines, and identified six sub-properties of a system's usability: robustness (likelihood of user error and ease with which users can correct errors), task efficiency, reusability of knowledge, effectiveness of user-computer communication, comprehensibility and flexibility.

Most of these principles also apply in the construction arena:

- Robustness—How easily can the structure be damaged? In heavy winds, will it be unroofed?
- Task efficiency—How easy is it to use the kitchen, for example. Is the distance between sink, oven and refrigerator optimized for the user? How far does one need to walk from the kitchen to the laundry?
- Flexibility—Can the construct be used for differing purposes? For example, can the building intended as a school gymnasium be used as a concert hall?
- Effectiveness of user-computer communication—Perhaps this is best thought of in terms of the ambience of the rooms or the building. It suggests the intention of the rooms. It could also be thought of in terms of the effectiveness of the acoustic design of an auditorium.
- Reusability of knowledge—This is less obvious, but could apply to the way the doors swing or fasten (consistency).
- Comprehensibility—Is the purpose of the design elements obvious? Are there nooks and crannies whose purpose defies analysis? Or does the layout "explain itself" as Heath describes?

In general, there are obvious usability parallels between the two disciplines.

## 4 Learning from history

For a traditional architect called on to design a hospital, it is important that every detail of the working of the hospital is known early. Space cannot be allocated or even the form of the buildings conceived until the people responsible for wards, kitchens and operating theatres, say, have been able to explain their requirements. While some of these people may know what they want, the architect taking a higher-level view may be able to suggest a better working arrangement. Similarly with software, task analysis can suggest potential for business process reengineering. Only after all the preliminary work is done for the hospital can an architect decide whether a small number of multi-storied buildings would suit better than a series of lower connected buildings. Up to this point there is no physical shape for the building, nor any architectural character (p.20 Richards (1974)). Richards further goes on to say that the visible form and the aesthetic quality for a building will emerge as part of the problem-solving, in contrast to the earlier approach of starting with a preconceived image and fixing the functionality to suit *(ibid.)*.

Writing about traditional architecture, Harrigan & Neel (1996) state: *“The way clients are looking to the future requires that we study our client’s situation more than we have ever done before. If we are to succeed, we must learn a great deal about how clients are organized and what strategies underlie their way of doing business.”* This is a quote about traditional architecture in modern times. A practising architect described it this way: *“Architects view design as a setting for human and “social” activities and hence are forced to deal with design in the context of its use. One trait of really gifted architects is to not only generate designs that clients aspire to but also to fulfill aspirations that they may not otherwise conceive of.”*

## 4.1 Doing it better in software

The situation is not dissimilar for software. To come up with a suitable conceptual design, one must be grounded in the underlying technology. Producing a conceptual design in the absence of detailed information about the underlying technology is fraught with danger. The author has observed projects where a conceptual design was delivered, but the technology eventually used for the development did not support a number of the facilities planned. Thus “work-arounds” resulted. Having one person responsible for carrying through the conceptual design is a necessary, but not sufficient, condition for success. The technical feasibility must be considered as well.

Conversely, proposing a solution without fully understanding the user interface requirements leads to problems. The author was involved in a “lessons learned” case study of a sophisticated multi-user computerised training system with a total hardware and software cost of approximately A$20M. More detailed descriptions of the project and the lessons learned are given in Schmidt et al. (1999). One particular problem was that a lack of consultation with the client, and end-users especially, led to the contractor developing an entirely inappropriate user interface.

The former example is analogous to a traditional architect renovating a 1800s warehouse, but using the old plumbing just where it was. The latter is like building a house to “normal” standards of door and bench heights, but not finding out until later that the owner-occupier of the house was more than 2.2 metres tall. The effective integration of usability into the software development process needs both user domain knowledge and technical expertise.

## 4.2 An iterative approach

There are many proponents of iterative approaches to software development, with the WinWin spiral model (Boehm et al., 1997) being touted frequently by software engineers. Usability consultants and HCI lecturers, on the other hand, swear by the Usability Cycle. Thus there is general agreement on the need to have an iterative process so that we can refine our designs for a usable product. Architects do sketches and build scale models to test out potential solutions. Software developers do not have the luxury of being able to scale up their models, so need to take a different approach. Prototypes can be used at varying levels of fidelity from paper to executables. They can be used to test out user interactions, or to test potential algorithms. However, final user testing is still imperative. Is there some way this repetitive process can be shortened?

Where houses are constructed by speculative builders they are using well-known patterns of construction which need little adaptation. Similarly for well-defined, well-practiced solutions to relatively small problems in software, probably a specialist architect is an overkill. The theme of patterns advocated by Alexander and others in traditional architecture (see, e.g. (Alexander et al., 1977)) has been taken up in a number of different software areas, including architectural design, usability and code ((Bass et al., 1998; Mahemoff & Johnston, 1998; Gamma et al., 1995)). Patterns promise to assist in reuse in many different ways, and may offer some help towards conceptual unity if the pattern language is tightly constructed.

Usability patterns in particular offer the opportunity to reduce the number of iterations required, as they embody solutions which have been proven to work, and which have already been through the repetitive process. It should be noted that usability patterns are more akin to Alexander's original idea of patterns than are the commonly-used object oriented patterns. Alexander had the user perspective in mind with his "habitable environment". Sometimes design rationale relates to a negative concept, especially in an area like safety critical design, where the designer tries to avoid the situatons that have been problematic in the past. Usability patterns, on the other hand, offer the opportunity to capture best practice—solutions that work!

Pattern languages offer an even better opportunity to capitalise on the existing investment, as member patterns have been generated from a particular design philosophy. Thus if software has been developed using a pattern language, it should be straightforward to apply related patterns when making changes to the software during maintenance.

## 5 Conclusions

There are many parallels which can be drawn between the situations confronting traditional architects and software developers. In particular, usability patterns can help capture good practice, so that there is a reduced need to iterate to produce an acceptable product.

We are also moving to a time where the "joy" and aesthetics of user interfaces are becoming more important. Are there lessons applicable here, which we can learn from traditional architecture?

## References

Alexander, C., Ishikawa, S. & Silverstein, M. (1977), *A Pattern Language*, Oxford University Press.

Bass, L., Clements, P. & Kazman, R. (1998), *Software Architecture in Practice*, SEI Series in Software Engineering, Addison Wesley Longman, Inc.

Boehm, B., Egyed, A., Kwan, J. & Madachy, R. (1997), "Developing Multimedia Applications with the WinWin Spiral Model", *ACM SIGSOFT Software Engineering Notes* **22**(6), 20–39.

Brooks Jr., F. P. (1995), *The Mythical Man-Month*, 20th anniversary edition, Addison-Wesley.

Gamma, E., Helm, R., Johnson, R. & Vlissides, R. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, MA.

Harrigan, J. E. & Neel, P. R. (1996), *The executive architect: transforming designers into leaders*, John Wiley.

Heath, T. (1991), *What, if anything, is an architect?*, Architecture Australia Pty.Ltd.

Kostof, S. (1985), *A History of Architecture*, Oxford University Press.

Kostof, S. (ed.) (1977), *The Architect*, Oxford University Press.

Mahemoff, M. J. & Johnston, L. J. (1998), Principles for a Usability-Oriented Pattern Language, *in* P. Calder & B. Thomas (eds.), *OZCHI '98 Proceedings*, IEEE Computer Society, Los Alamitos, CA, pp.132–139.

Pressman, R. S. (1997), *Software Engineering: A Practitioner's Approach*, 4th edition, McGraw-Hill Inc.

Richards, J. (1974), *Architecture*, The Professions, Newton Abbot, David and Charles.

Schmidt, C., Dart, P., Johnston, L., Sterling, L. & Thorne, P. (1999), "Disincentives for Communicating Risk: A Risk Paradox", *Information and Software Technology* **41(7)**, 403–411.

Stevens, R. (1965), *Building in History*, Cassell, London.

# Evidence-Based Usability Engineering: Seven Thesis on the Integration, Establisment and Continuous Improvement of Human-Centred Design Methods in Software Development Processes

**Eduard Metzker**

DaimlerChrysler Research & Technology, Software Technology Lab,

Wilhelm-Runge Str. 11, P.O. Box 2360, D-89013 Ulm, Germany

eduard.metzker@daimlerchrysler.com

**Abstract:** In this position paper we propose an approach for the systematic integration of human-centred design (HCD) methods in software development process, called evidence-based usability engineering. Instead of clinging to a fixed workflow model of the usability engineering process our approach advocates a paradigm of situated decision making to enable development teams to select an optimal set of HCD methods based on the available evidence of the engineering task at hand and the experience of the software development organization. Our approach is linked to process assessment tools such as UMM (Usability Maturity Model) via a meta-model that guides the introduction, establishment and continuous improvement of HCD methods and promotes organizational learning in HCD. We present first concepts of a novel kind of process-centred usability engineering environment, ProUSE, to support our evidence-based usability engineering approach. We develop our approach by proposing seven thesis that emphasize shortcomings and urgent requirements of current HCD practice and that are based on our recent research efforts and experiences with HCD process improvement at DaimlerChrysler.

**Keywords:** usability maturity, process improvement, human-centred design methods

## 1 Introduction

The relevance of usability as a software quality factor is continually increasing for software development organizations: usability and user acceptance are about to become the ultimate measurement for the quality of today's mobile services and tomorrows proactive assistance technology. Taking these circumstances into account human-centered design (HCD) methods for designing interactive systems are changing from a last minute add-on to a crucial part of the software development lifecycle.

It is well accepted both among software practitioners and in the human-computer interaction research community that structured approaches are required to build interactive systems with high usability. On the other hand specific knowledge about exactly how to most efficiently and smoothly integrate HCD methods into established software development processes is still missing [1]. While approaches such as the usability maturity model (UMM) [2] provide means to assess an organizations capability to perform HCD processes it lacks guidance on how to actually implement process improvement in HCD. It often remains unclear to users of HCD methods if and why certain tools and methods are better suited in a certain development context than others [3]. In fact we know very little of the actual value of the methods that we propose. We lack strategies and tools that support development organizations in evaluating and selecting an optimal HCD method for a given development context and perform systematic process improvement in HCD. Little research has been done on integrating methods and tools of HCD in the development process and gathering knowledge about HCD activities in a form that can capture relationships between specific development contexts and applicable methods and tools [4].

In this paper we work out seven thesis that point out shortcomings and requirements of current HCD practice. First we take a look at existing HCD process models, trying to identify the organizational obstacles that hamper the establishment of these models in mainstream software development processes. Next we present results of a survey where we examined how exactly HCD methods are applied in actual projects and derived implications for tools to support process improvement in HCD. Finally we outline the concepts of an evidence-based usability engineering approach that we propose to address the shortcomings and requirements we have come across.

## 2 Existing HCD process models

There is a large body of research and practical experience available on software process models which are used to describe and manage the development process of software systems. Prominent examples are the *waterfall model* [5], the *spiral model* [6], or the *fountain model* [7]. Yet for the shortcomings of traditional process models concerning usability issues, a number of approaches have been developed that take into account the special problems encountered with the development of highly interactive systems [1, 8-11]. According to ISO13407 these approaches can be embraced by the term ‘human-centered design processes’ [12]. In this section we focus on those approaches which have been extensively applied to industrial software development projects. We outline their basic principles and discuss some of their drawbacks based on documented case studies.

One of the first approaches used to address usability issues was the soft system methodology (SSM) [8, 13]. SSM was widely applied to capture the objectives, people involved (e.g. stakeholder, actors, and clients), constraints, and different views of interactive systems during development. However, since SSM’s origins are in general systems theory, rather than computer science, it lacks many of the specific HCD activities such as construction of *user interface mockups* or iterative *usability testing* which are necessary to fully specify interactive systems. These shortcomings limit the utilization of SSM to the early activities of the development process such as requirements or task analysis.

The star lifecycle [9], proposed by Hix and Hartson, focuses on usability evaluation as the central process activity. Around this central task the, activities *system / task / functional / user analysis*, *requirements / usability specifications*, *design & design representation*, *rapid prototyping*, *software production and deployment* are placed. The results of each activity such as task analysis are subjected to an evaluation before going on to the next process activity. The bi-directional links between the central usability evaluation task and all other process activities cause the graphical representation of the model to look like a star.

One problem concerning this approach was already outlined by Hix and Hartson [9]: Project managers tend to have problems with the highly iterative nature of the model. They find it difficult to decide when a specific iteration is completed, complicating the management of resources and limiting their ability to control the overall progress of the development process. Furthermore, the star lifecycle addresses only the interactive parts of a software system, leaving open how to integrate the star lifecycle with a general software development method.

The *usability engineering lifecycle* [1] by Deborah Mayhew is an attempt to redesign the whole software development process around usability engineering knowledge, methods, and activities. This process starts with a structured requirements analysis concerning usability issues. The data gathered from the requirements analysis is used to define explicit, measurable usability goals of the proposed system. The *usability engineering lifecycle* focuses on accomplishing the defined usability goals using an iteration of usability engineering methods such as *conceptual model design*, *user interface mockups*, *prototyping* and *usability testing* [10]. The iterative process is finished if the usability goals have been met.

As outlined by Mayhew [1], the usability engineering lifecycle has been successfully applied throughout various projects. However, some general drawbacks have been discovered by Mayhew during these case studies: One important concern is that redesigning the whole development process around usability issues often poses a problem regarding the organizational culture of software development organizations. The well established development processes of an organization can not be turned into human-centered processes during a single project. Furthermore, the knowledge necessary to perform the HCD activities is often missing in the development teams, hampering the persistent establishment of HCD activities within the practiced development processes. How the HCD activities proposed in the usability engineering lifecycle should be integrated exactly and smoothly into development processes practiced by software development organizations, was declared by Mayhew as an open research issue [1].

*Usage-centered design* [11], developed by Constantine and Lockwood, is based on a process model called *activity model for usage-centered design*. The activity model describes a concurrent HCD process starting with the activities *of collaborative requirements modeling, task modeling,* and *domain modeling*, in order to elicit basic requirements of the planned software system. The requirements analysis phase is followed by the design activities: *interface content modeling* and *implementation modeling*. These activities are continuously repeated until the system passes the usability inspections carried out after each iteration. The design and test activities are paralleled by *help system / documentation development* and *standards / style definition* for the proposed system. This general framework of activities is supplemented by special methods like *essential use case models* or *user role maps*.

Constantine and Lockwood provide many case studies where usage centered design was successfully applied, yet they basically encountered the same organizational obstacles as Mayhew [1] when introducing their HCD approach into software development processes practiced. They emphasize the fact that ‘new practices, processes, and tools have to be introduced into the organization and then spread beyond the point of introduction’ [11]. A straightforward solution to these problems is training courses for all participants of HCD activities offered by external consultants. However, this solution is regarded as being time consuming and cost intensive in the long run. It tends to have only a limited temporary effect and thus does not promote organizational learning in HCD design methods [11]. Constantine and Lockwood conclude that it is necessary to build up an internal body of knowledge concerning HCD methods, best practices and tools tailored to the needs of the development organization.

The organizational obstacles that are encountered in establishing HCD methods in development processes can be summarized in the claims 1-4 [14]:

**Claim 1** *Existing HCD process models are decoupled from the overall software development process.*

One common concern relating to HCD approaches is that they are regarded by software project managers as being somehow decoupled from the software development process practiced by the development teams. It appears to project managers that they have to control two separate processes: the overall system development process and the HCD process for the interactive components. As it remains unclear how to integrate and manage both perspectives, the HCD activities have often been regarded as dispensable and have been skipped in case of tight schedules [1].

**Claim 2** *Existing HCD approaches assume that HCD methods can be performed by the development team ad hoc.*

Most approaches also assume that experienced human factors specialists are available throughout the development team and that HCD methods can be performed ad hoc. However, recent research shows that even highly interactive systems are often developed without the help of in-house human factors specialists or external usability consultants [15]. Therefore HCD methods often can not be utilized because the necessary knowledge is not available within the development teams [[1, 2]]

**Claim 3** *Existing UE process models are not tailorable to the usability maturity of software development organizations.*

Another point that is also ignored by the approaches described is that development organizations are often overwhelmed by the sheer complexity of the proposed HCD process models. The models lack a defined procedure for tailoring the development process and methods for specific project constraints such as system domain, team size, experience of the development team or the system development process already practiced by the organization.

**Claim 4** *Integrating UE Methods into mainstream software development process must be understood as an organizational learning task.*

Almost all approaches do not account for the fact that turning technology-centered development processes into human-centered development processes must be seen as a continuos process improvement task [16]. A strategy for supporting a long-lasting establishment of HCD knowledge, methods, and tools within development organizations is still missing. A model is needed that guides the introduction, establishment and continuous improvement of UE methods in mainstream software development processes.

## 3 Human-Centred Design in Practice

To compare these findings with To be able to construct a tool for the effective support of UE processes, we needed in-depth knowledge of the future users of such a tool and their requirements. This led to the following central questions:

- What kind of development process for interactive systems is practiced by the development organizations in their projects?
- What typical tasks do the developers have to solve?
- What problems are typical for the development process?
- What are possible implications for tool support?

The survey was elaborated, performed and evaluated in collaboration with industrial psychologists and had the following structure [17]: A questionnaire was used to record both personal data and information on the respondents' professional experience and typical development tasks. A semi-structured interview supplemented by a special set of questions concerning the application of UE activities during the development process as perceived by the respondents formed the core of the survey.

A total of 16 employees from four major companies[i] involved in the development of interactive software systems were selected. The respondents are engaged in developing these systems in projects from diverse domains: military systems, car driver assistance technology or next-generation home entertainment components. The questioning was performed by a single interviewer and the answers were recorded by a second person in a pre-structured protocol document. Each interview took between 90-150 minutes.

The organizations examined are practicing highly diverse individual development processes, however non of the UE development models proposed by [1, 10, 11, 18] are exactly used.

Furthermore, the persons who are entrusted with the ergonomic analysis and evaluation of interactive systems are primarily the developers of the products. External usability or human factors experts or a separate in-house ergonomics department are seldom available. Furthermore, few of the participants were familiar with basic methods like *user profile analysis* or *cognitive walkthrough*.

The UE methods that are considered to be reasonable to apply by the respondents are often not used for the following interrelated reasons:

- There is no time allocated for UE activities: they are neither integrated in the development process nor in the project schedule.
- Knowledge needed for the performance of UE tasks is not available within the development team.
- The effort for the application of the UE tasks is estimated to be too high because they are regarded as time consuming.

### 3.1 Survey Conclusions

The results of the survey led to the following conclusions regarding the requirements of a software tool to support the improvement of UE processes:

**Claim 5** *Support flexible UE process models*

The tool should not force the development organization to adopt a fixed UE process model as the processes practiced are very diverse. Instead, the tool should facilitate a smooth integration of UE methods into the individual software development process practiced by the organization. Turning technology-centered processes into human-centered processes should be seen as a continuous process improvement task where organizations learn which of the methods available best match certain development contexts, and where these organizations may gradually adopt new UE methods.

**Claim 6** *Support evolutionary development and reuse of UE experience*

It was observed that the staff entrusted with ergonomic design and evaluation often lacks a special background in UE methods. Yet, as the need for usability was recognized by the participating organizations, they tend to develop their own in-house usability guidelines and heuristics. Recent research [19-22] supports the observation that such usability best practices and heuristics are, in fact, compiled and used by software development organizations. Spencer [21], for example, presents a streamlined cognitive walkthrough method which has been developed to facilitate efficient performance of cognitive walkthroughs under the social constraints of a large software development organization. However, from experiences collected in the field of software engineering [23] it must be assumed that, in most cases, best practices like

[i] DaimlerChrysler Aerospace (DASA) in Ulm, Sony in Fellbach, Grundig in Fuerth and DaimlerChrysler in Sindelfingen (all sites are located in Germany)

Spencer's are unfortunately not published in either development organizations or the scientific community. They are bound to the people of a certain project or, even worse, to one expert member of this group, making the available body of knowledge hard to access. Similar projects in other departments of the organization usually cannot profit from these experiences. In the worst case, the experiences may leave the organization with the expert when changing jobs. Therefore, the proposed tool should not only support existing human factors methods but also allow the organizations to compile, develop and evolve their own approaches.

**Claim 7** *Provide means to trace the application context of UE knowledge*

UE methods still have to be regarded as knowledge-intensive. Tools are needed to support developers with the knowledge required to effectively perform UE activities. Furthermore, the tool should enable software development organizations to explore which of the existing methods and process models of UE works best for them in a certain development context and how they can refine and evolve basic methods to make them fit into their particular development context. A dynamic model is needed that allows to keep track of the application context of UE methods.

## 4 The Evidence-Based Usability Engineering Approach

To address the shortcomings and meet the requirements described in our claims we advocate an evidence-based approach to the improvement of HCD processes.

The essence of the evidence-based approach is that we do not cling to a fixed workflow model of the usability engineering process, but instead follow a paradigm of situated decision making. In this approach HCD methods are selected based on the available evidence that they will match to the development context at hand.

After performing each method it should be evaluated if the method was useful for the development context or if it must be modified. The modification of the method should be recorded and stored as a best practice for later reuse. Once a certain body of HCD knowledge is accumulated in that way, we have sound evidence for selecting an optimal set of HCD best practices for given development context. Via continuously appyling this procedure of conducting, evaluating and adpting HCD methods, an organization gradually adapts a set of HCD base practices to a wide variety of development contexts. This directly contributes to the general idea of software maturity models such as UMM. According to these models organizations are highly ranked on a maturity scale, if they are capable to tailor a set of base practices according to a set of constraints such as available resources or project characteristics to achieve a defined engineering task.

So far we argue that the evidence based approach requires three ingredients:

- A process meta-model, which guides the selection of HCD methods for a given development context and their integration in an overall software development process in a flexible, decision-oriented manner. The model must as well provide a strategy for evaluating, refining and capturing best practices for new development contexts thus promoting continuous process improvement and organizational learning in HCD.
- A concept for an experience base that allows to keep track of documented best practices and their application context even if the underlying context factors such as processes, technologies, domains and quality standards are still evolving.
- A tool concept for managing the experience base and that allows to predict optimal sets of HCD method based on the available evidence of the engineering task and the experience of the development organization.

### 4.1 The Evidence-Based Improvement Model

Our evidence-based model for human-centered design processes comprises a set of organizational tasks that support the introduction, establishment and continuous improvement of HCD methods throughout the whole development lifecycle. It helps to manage and tailor the HCD base practices defined in UMM [2] and the related methods practiced by the development organization according to specific constraints of the respective project and the needs of the development organization. These organizational tasks are grouped in our model as depicted in Figure 1. The evidence-based usability engineering model is based on our findings of experience based improvement of HCD processes [14]. We refined the model to the extend that we now use the UMM as a basis for assessing which HCD base practices are conducted by an organization and which are missing. Then the selection of the actual HCD methods is guided by a model of the development context. So while the UMM framework is used to assess which HCD base practices should be conducted, a model of the development context is used to map the development context to an optimal set of HCD methods.

In more detail the model consists of the following four logical steps:

*Step 1: Analyze the HCD activities practiced*

The first step comprises an analysis of the practiced HCD process and the related HCD base practices, to elicit when, where and how HCD methods are performed within the software development lifecycle in use. The deliverable of this step is a documentation of possible improvements of the HCD process that is currently used. This assessment can be performed using an UMM questionnaire.

*Step 2: Select suitable HCD base practices and integrate them into the practiced software development process*

The results of the first step form the rationale for the selection of HCD base practices from UMM reference model for the improvement of the development process. The HCD base practices which have been selected for the improvement of the development process have to be integrated in the model of the practiced software development lifecycle and the project planning and form the improved development process.

However, in this step further important factors have to be considered, e.g. the type of system to be developed and project constraints like budget and schedules. This evidence must be mapped into a context model and guide the selection of appropriate HCD methods to perform the selected base practices.

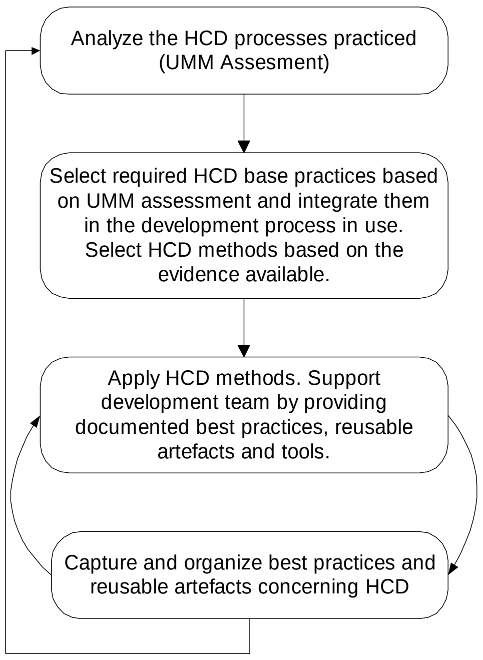

Analyze the HCD processes practiced (UMM Assesment)
Select required HCD base practices based on UMM assessment and integrate them in the development process in use. Select HCD methods based on the evidence available.
Apply HCD methods. Support development team by providing documented best practices, reusable artefacts and tools.
Capture and organize best practices and reusable artefacts concerning HCD

**Figure 1:** Steps of the evidence-based model

*Step 3: Support effective performance of the defined HCD methods*

Generally, at this step in the model resources have already been allocated for HCD activities, e.g., a usability engineer was nominated, who is responsible for coordinating and supporting the execution of the various HCD activities of the new process. However, the efficiency and impact of the proposed HCD methods must be increased by providing the development team with best practices, tools and reusable deliverables of past projects (e.g. templates for usability test questionnaires, results of conceptual task analysis or user interface mockups) which facilitate effective performance of the selected HCD methods. This set of information should be easily accessible for all participants of HCD activities.

*Step 4: Collect and disseminate best practices and artifacts concerning HCD tasks*

During the execution of HCD activities, artifacts with a high value for reuse in the same or subsequent projects are generated by the participants of HCD activities, for example, templates for usability tests, reusable code fragments, or an experience on how to most efficiently conduct a user profile analysis for assistence systems. Observation like this comprise HCD experience and rationale that have to be captured and organized in best practices along with the development context in which they apply to allow for easy reuse in the same or subsequent projects.

The evidence-based model contains two cycles: The inner cycle between step 3 and 4 supports the introduction and establishment of HCD activities and methods within the practiced software development process. It supports the effective utilization and improvement of HCD methods selected by fostering the application of best practices which are tailored to the needs of the development organization. This cycle is continuously iterated during the development process.

The outer cycle which connects step 4 and 1 should be performed in the ideal case at least twice during the development process of large projects as it serves the improvement of the overall HCD processes practiced by an organization.

### 4.2 The HCD Experience Base

To capture and evolve HCD knowledge for reuse and process improvement, we need a concept for an HCD experience base. For this purpose we have developed the concepts of USEPACKs (Usability Engineering Experience Package) and a context model. While a USEPACK is used to capture HCD best practices a context model is used to formally relate these best practices to a development context.

*The USEPACK concept*

A USEPACK is a semi-formal notation for structuring knowledge relating to HCD activities. It encapsulates best practices on how to most effectively perform certain HCD activities and includes the related artifacts like documents, code fragments, templates and tools that facilitate the compliance with the best practice described. A USEPACK is structured into five logical sections:

- The *core information* permits authors to describe the main message of a USEPACK. It is organized according to the pyramid principle for structuring information [24]. The information first presented to the reader has a low level of complexity, allowing the reader to quickly decide if the USEPACK is worth further exploration. With further reading, the degree of complexity rises, introducing the reader to the experience described. The core information section includes the fields *title*, *keywords*, *abstract*, *description* and *comments*.
- The *context situation* describes the development context related to the experience in question. The context situation is generated by using the context model, allowing the authors and readers of USEPACKs to utilize a shared vocabulary for contextualizing and accessing USEPACKs.
- A set of *artifacts*, such as checklists for user profile analysis or templates for usability test questionnaires, facilitates the efficient compliance with the best practice. They represent an added value to the readers of a USEPACK. Artifacts allow readers to regain time spent on exploring the package by using the supplied artifacts to simplify their work.

### 4.3 The Context Model Concept

The context model serves as a template to construct the *context situation* for USEPACKs – a semi-formal description of the development context in which the information of a USEPACK can be applied. It is organized in a hirarchical structure, divided into sections which contain groups of *context factors*. On the one hand, authors can use the context model to easily construct a description of the context in which the information of a USEPACK can be applied by selecting appropriate context factors from the model. On the other hand, readers can use the context model to specify a context situation which reflects the development context for which they need support in the form of USEPACKs. Currently a context model containing the following four sections is used:

- The *process context* section provides context factors to describe to which base practices of the UMM reference model a best practice is related.
- The *project context* section provides context factors to describe project constraints like the size of the development team, budget or project duration which are related to the experience cited.
- The *domain context* section provides context factors to describe elements of the domain related to the experience described. Top-level context factors of this section specify domains in terms like ‘home entertainment systems’ or ‘car driver assistance systems’, which can be subsequently refined to capture more detailed domain attributes.
- The *technology context* section provides context factors to describe features of technologies related to the experience described like ‘gesture recognition’ or ‘speech input’.

### 4.4 Tool Support

To increase the impact of the evidence-based usability engineering approach tool support is needed. In the BMBF[ii] lead project EMBASSI[iii] we currently develop a prototypical tool called ProUSE (Process centred Usability Engineering Environment). ProUSE consists of an HCD experience base and three logical components as depicted in Figure 2.

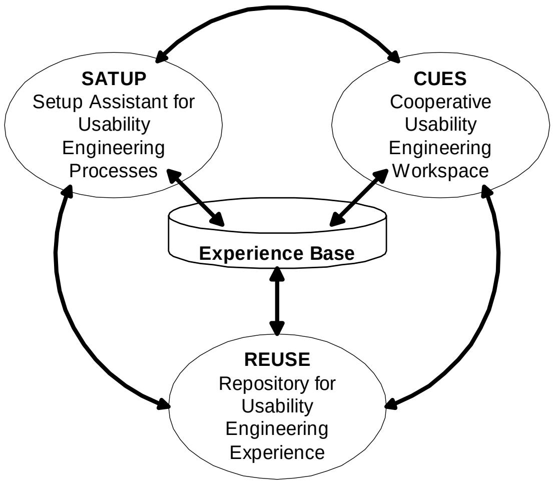

SATUP
Setup Assistant for Usability Engineering Processes
CUES
Cooperative Usability Engineering Workspace
Experience Base
REUSE
Repository for Usability Engineering Experience

**Figure 2 :** Logical Components of ProUSE

The experience is seeded with an initial set of best practices in form of USEPACKs. In our case we have adopted a variety of usability engineering methods from Nielsen and Mayhew[1,10] but in general any HCD approach should be appropriate.

The REUSE (Repository for Usability Engineering Experience) [15] component is used to capture, manage and evolve best practices related to HCD activities. It assists in documenting best practices using the USEPACK concept and relating them to a formal development context using a context model and storing them in the experience base.

SATUP (Setup Assistant for Usability Engineering Processes) is used to plan HCD activities for a software development project. Given all available context information on the process (e.g. which HCD base practices should be performed), project (e.g. duration, budget, team size), domain and technology context factors, SATUP will propose optimal HCD methods and reusable artefacts based on the accumulated experience of the development organization.

Once an optimal HCD process was planned, CUES (Cooperative Usability Engineering Workspace) can be used by a distributed development team to perform the HCD methods selected.

The ProUSE prototype is based on Java technologies and integrated via a web portal concept which makes the modules available through intranet and web browser.

First versions of SATUP, CUES and REUSE are currently evaluated with our consortium partners[i] so that we expect some interesting results soon.

## 5 Discussion

The findings, concepts and tools presented in this paper reflect the experiences we collected in the recent years with improving HCD processes in our business units at DaimlerChrysler and indicate research directions we currently explore. Hopefully the vast amount of ideas provides a rich foundation to stimulate further discussions in directions such as:

- What concepts exist for knowledge/experience based usability engineering approaches and tools?
- We care a lot about usable systems, but what can we do to make our methods usable for development teams?
- How can we more reliably evaluate the actual value of the HCD methods we propose?
- Is there an optimal form for capturing HCD knowledge that balances the needs for ease of use and formality?
- How do we balance that needs for structured approaches and creativity in HCD approaches?

## 6 References

1. Mayhew, D.J., *The Usability Engineering Lifecycle: A Practioner's Handbook for User Interface Design*. 1999: Morgan Kaufman.
2. Earthy, J. *Human Centred Processes, their Maturity and their Improvement*. in *IFIP TC.13 International Conference on Human-Computer Interaction*. 1999. Edinburgh, UK: British Computer Society.
3. Welie, M.v. *Breaking Down Usability*. in *IFIP TC.13 International Conference on Human-Computer Interaction*. 1999. Endinburgh, UK: IOS Press.
4. Henninger, S., *A Methodology and Tools for Applying Context-Specific Usability Guidelines to Interface Design*. Interacting with Computers, 2000. **12**(3): p. 225-243.

[ii] German Ministry of Education and Research

[iii] Electronic Multimedia Operating and Service Assistence

5. Royce, W.W. *Managing the development of large software systems*. in *IEEE WESTCON*. 1970. San Francisco, USA.
6. Boehm, B.W., *A spiral model of software development and enhancement.* IEEE Computer, 1988. **21**(5): p. 61-72.
7. Henderson-Sellers, B. and J.M. Edwards, *Object-Oriented Systems Life Cycle.* Communications of the ACM, 1990. **31**: p. 143-159.
8. Checkland, P.B., *Systems Thinking, Systems Practice*. 1981: John Wiles & Sons.
9. Hix, D. and H.R. Hartson, *Iterative, Evaluation-Centered User Interaction Development,* in *Developing User Interfaces: Ensuring Usability Through Product & Process*. 1993, John Wiley & Sons: New York. p. 95-116.
10. Nielsen, J., *Usability Engineering*. 1994: Morgan Kaufman Publishers.
11. Constantine, L.L. and L.A.D. Lockwood, *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. 1999: Addison-Wesley.
12. ISO/TC 159 Ergonomics, *Human-centered Design Processes for Interactive Systems,* . 1999, ISO International Organization for Standardization.
13. Checkland, P.B. and J. Scholes, *Soft Systems Methodology in Action*. 1990: John Wiley & Sons.
14. Metzker, E. and M. Offergeld. *An Interdisciplinary Approach for Successfully Integrating Human-Centered Design Methods Into Development Processes Practiced by Industrial Software Development Organizations. accepted for: 8th IFIP Working Conference on Engineering for Human-Computer Interaction, EHCI2001*. 2001. Toronto Canada
15. Metzker, E. and M. Offergeld. *REUSE: Computer-Aided Improvement of Human-Centered Design Processes*. in *Mensch und Computer, 1. Fachübergreifende Konferenz, MC2001*. 2001. Bad Honnef, Germany: Teubner Verlag.
16. Norman, D.A., *The Invisible Computer*. 1998: MIT Press.
17. Wetzenstein, E. and A. Becker, *Requirements of Software Developers for a Usability Engineering Environment,* . 2000, Artop Institute for Industrial Psychology: Berlin.
18. Beyer, H. and K. Holtzblatt, *Contextual Design: Defining Customer-Centered Systems*. 1998: Morgan Kaufmann.
19. Weinschenk, S. and S.C. Yeo, *Guidelines for Enterprise-wide GUI design*. 1995, New York: Wiley.
20. Billingsley, P.A., *Starting from Scratch: Building a Usability Programm at Union Pacific Railroad.* Interactions, 1995. **2**(4): p. 27-30.
21. Spencer, R. *The Streamlined Cognitive Walkthrough Method: Working Around Social Constraints Encountered in a Software Development Company*. in *CHI2000*. 2000. The Hague: ACM Press.
22. Rosenbaum, S., J.A. Rohn, and J. Humburg. *A Toolkit for Startegic Usability: Results from Workshops, Panels and Surveys*. in *Conference on Human Factors in Computing Systems*. 2000. The Hague, Netherlands: ACM press.
23. Basili, V.R., G. Caldiera, and H.D. Rombach, *Experience Factory*, in *Encyclopedia of Software Engineering*, J.J. Marciniak, Editor. 1994, John Wiley & Sons: New York. p. 528-532.
24. Minto, B., *The Pyramid Principle - Logic in Writing and Thinking*. 3 ed. 1987, London: Minto International Inc.

# User Intelligence Will Make Mobile Solutions Fly

Anna Olsson, Razorfish AB
anna@razorfish.com

Sofia Svanteson, Ocean Observations
sofia@oceanobservations.com

Stockholm, Sweden, April 2001

## Introduction

Consideration of user intelligence is critical when developing solutions for mobile applications or products, and is a necessary and effective way to tackle many of the challenges inherent in mobility. Usability is a qualitative focus one takes when defining the user experience, while usefulness must be acquired through performing user research, i.e. user intelligence, on the presumptive users. Creating a thorough user experience for a product or service requires a number of activities to be carried out to achieve the desired qualitative standard. Without aiming for usefulness, the usability can be excellent but still worthless.

Understanding users is the true foundation for developing an effective business strategy and digital experiences, which connect wired or wireless ventures with their users in meaningful ways. While many firms say they “design for the user experience”, what they usually mean is that they refine the site architecture and navigation for an optimal on-line experience subsequent to prototype development.

*The user experience does not begin or end with the digital interface.*

The greatest potential for understanding the user lies in capitalizing on this knowledge to set the strategic course up-front, prior to concept development. This means understanding user needs and preferences on and off-line, the relationship between the digital realm and the non-digital realm in users’ lives, the boundaries between work and the rest of life, and finally their situations, needs, behaviors, values and skills. Expectations and standards for a particular digital experience are crafted with regard to interactions in every other aspect of life and work. Therefore, identifying unmet user needs, and determining how services and products can add significant value to their lives, must be determined with respect to the larger context of their lives. Only when users are understood in context, can an effective strategy for connecting with them digitally be developed. This effective strategy is called “User Intelligence”.

This paper highlights some of the challenges found within mobile solutions and illustrates how attention to user intelligence can improve the way services are built and thus the way people interact with each other and with mobile devices.

## The New Mobile Culture

Wireless technology is on its’ way to radically alter social behaviors and patterns. The affordability of mobile phones has enabled many more people than before to be mobile. As a result, greater numbers of people worldwide are free of the physical, economic, and organizational constraints associated with wired communication. Mobility used to be a privilege reserved primarily for the wealthy, now most people in the western world can be mobile. The result is a new society characterized by increased freedom, independence, and the power to think and work beyond traditional confines of space and time.

Understanding that mobility has reshaped our society leads to the next point: the new mobile citizenry has developed its own distinctive culture with its own etiquette and behaviors that did not exist prior to wireless technology. Like most cultures, mobile culture has evolved, and will continue to do so, over time, largely driven by the current parameters of wireless infrastructure, devices, and software. Although technology made this new culture possible, one has to realize that a culture needs to be nurtured and paid attention to. The wireless technology cannot be developed in a vacuum where there is no anchoring in the real world and the common people. Today, there are several proofs of this vacuum. The mobile industry in Europe and the US is to a large extent governed and developed by engineers who have not paid sufficient attention to the target groups of mobile devices. Mobile phones are not esthetically pleasing enough, navigation is tricky and services are hard to use. There are several network and device constraints. The circuit-switched network makes interactions within a mobile service time consuming. Mobile devices have no common standards for design and functionality. Furthermore, screens are tiny, resolution and data entry is poor and these factors make it even harder for designers to create services for these devices. Having in mind the performance of mobile devices, and the desires of the target group, is essential when developing mobile solutions.

## User Intelligence

A mobile user intelligence approach should be formed in order to analyze the situations, needs and behaviors of possible users in a mobile context and thus build mobile solutions that meet their immediate needs within that context. One can study the mobile culture with help from deep interviews, contextual interviews, video ethnography, quantitative analysis, visual stories, life stage comparison, secondary research etc. The user analysis consists of two steps; frame findings and gain insight.

Frame findings is about making structure and find patterns of:

- **People** – who are they?
- **Activities** – what do people do?
- **Places** – where do they do it?
- **Time** – when or how often do they do it?
- **Tools** – what helps them do it?
- **Interactions** – how do they do it?

Gain insights is about underlying:

- **Goals** – why do they do it?
- **Motivations** – what makes them do it?
- **Problems** – what problems are there?
- **Difficulties** – what do they have to deal with?
- **Met/Unmet needs** – what do they need?
- **Desires** – what do they really want?
- **Values** – what does it mean to them? What is meaningful to them?

## Benefits of Performing User Intelligence

Provides input to business strategy and site design to ensure optimal user experience.

- Describes intended users in a meaningful way, and highlights significant differences, which affect product design (gender, life stage, values, capabilities, interests).
- Provides information for the underpinnings of brand strategy.
- Identifies unmet user needs and points the way to strategic opportunities for adding value digitally.
- Provides essential information and inspiration for designers.
- Identifies and prioritizes meaningful and essential content.
- Provides direct implications with respect to revenue generation – m-commerce (expectations, values) and appropriate advertising (synergies, partnerships, sponsors).
- Minimizes random or capricious architecture or visual design changes late in the game.

## Case Study: Mobile Buddies[1]

In the Mobile Buddies project user intelligence was applied in the form of deep interviews and contextual interviews in order to find out in what way people would like to communicate with, and locate, their friends with help from utilizing different platforms.

The deep interviews took place in a comfortable room where one observer, one moderator and the subject were the only present. The contextual interviews were set in a mobile context, in a public environment, where the subjects were likely to use their mobile phones. These public environments were located in the city center during lunchtime, popular bars on Thursday-Saturday nights, and café neighborhoods during Saturday and Sunday afternoons. The locations were selected due to an email research session within the target group. A total of 30 subjects participated in the interviews.

The questions asked in the interviews focused on topics such as:

- How many mobile phone calls do you make during lunchtime/evenings/weekends and what is the purpose of these calls?
- How many SMS:es do you send every week? Why and when do you send them?
- In what situations do you want to know where your friends are located?
- In what situations is it all right that your friends know where you are located? Why?
- How often do you use the Yellow Pages or different city guides? Why do you use these information sources?
- Describe your habits during lunchtime. How do you find someone to have lunch with?
- Describe your habits on weekday nights. How do you plan your evenings? How do you find your friends? Can this process be improved somehow?
- Describe your habits during the weekends? How do you plan your day? How do you find your friends? Can this process be improved somehow?

The results that came out of the study showed us, among other things, that:

- Users want to know where their friends are especially around lunchtime during weekdays and Friday and Saturday nights.
- Users want “visibility” to be optional.
- Users want to be able to create different groups of friends, coworkers etc and give them different rights.
- Friends are the main source when looking for some “entertainment”. Word of mouth is more reliable than city guides.
- Users want to be alerted when they are passing the bank, post office, pharmacy etc since they often forget to go to these institutions during lunch hours although they are in the near hood…
- They also want to be alerted when friends are at particular places during certain time slots.

## Conceptualization

Conceptualization is about creating consistent user and brand experiences through features, functions, content and design across appropriate platforms.

[1] Mobile Buddies is an instant messaging style communication solution for WAP, SMS, DTMF and WWW equipped with positioning. The solution is aimed at the private consumer market. The service is bundled with an operator’s positioning service and subscription. Mobile Buddies enables access to a community whenever and wherever the user feels like it. Razorfish AB in Stockholm, Sweden created Mobile Buddies.

In this phase the users' situations are thoroughly analyzed to gain a deeper understanding of the users' needs and how these can be transformed into creative concepts and desired products and services.

The conceptualization is a way to package and profile the functionality. This is also done to align the users' goals with the client's business goals and brand values. In this way a more evolved and finely tuned user experience can be delivered, which gives that added value to the intended users, which in turn, can create the basis for having satisfied customers and the foundation to a strong customer relationship. That is what in the end will differentiate products and services against their competitors in the marketplace. A well-substantiated relationship, grounded in user intelligence studies is difficult to copy.

The methods and tools used for Mobile Buddies in the Conceptualization phase are presented below:

- **P n' Func** – a matrix that shows how the users' needs and wanted experiences match with the defined functionality, features, content, business goals and brand values.
- **The Flop** – a grading tool for how important each function or feature is to every user profile.
- **Nomatrix** – a tool that help us combine and develop features in a creative way.
- **Geo time** – a matrix that helps mapping out, time, place and environment, which are important parameters to keep track of in the mobile world.
- **Mood boards** – proposals for the visual expression.

After having run the user research material through the conceptualization program we came up with a number of functions and features. The following are a few examples of the Mobile Buddies functionality:

**Send a message to all buddies within an area**
The subjects expressed support for an efficient "group communication" when looking for friends who are in a certain mood and or location.

**Search buddy**
A buddy does not answer his or her phone and you want to find out where he or she is. (It is probably Friday night and the buddy is at a noisy bar.)

**Find Bookmark**
There was a great interest in getting useful information from your buddies about a city area one does not know too well.

**Alert me!**
The subjects thought it would be useful to know when some of their buddies were located at the same place or in the same area.

## Prototyping

By the use of prototyping techniques the results are visualized in form as well as in functionality. This makes the decision-making process easier and reduces the uncertainty that may come with the gap between the conceptualization and the finalized product or service.

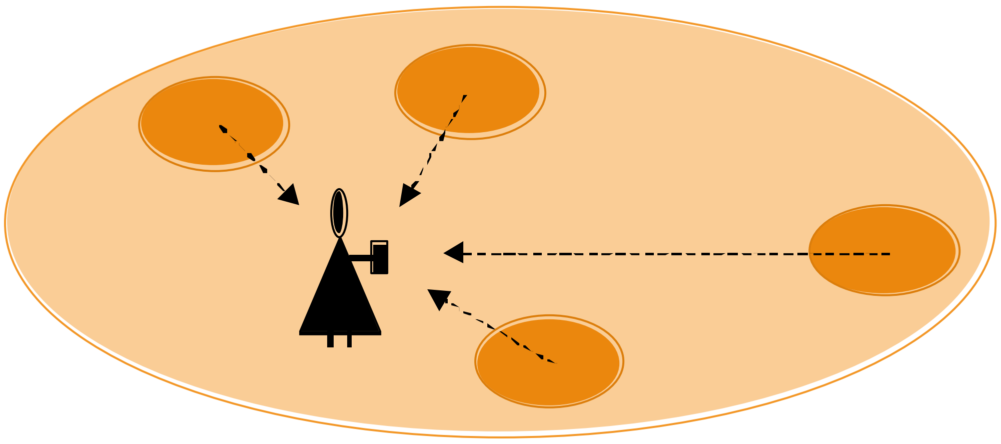
The differentiating factor for a product or service's success is the user's experience. It is therefore obvious to have both the users' experience and the client's business goals in mind when designing. With the aid of prototyping, a dialog can be kept with the users even before the product is implemented. That guarantees that it can live up to the demands held by the target audience and that we will produce products and services that are both usable and useful.

## User Scenarios

The purpose of creating user scenarios originates in the view of having a user focus throughout the whole design process. In order to create desired user experiences it is important to find out where the user is, why is the user there, how does the user feel, what does the user need in order to feel helped, relaxed and more efficient. By using the information, gathered from user interviews and other material, when creating future user scenarios one gets a better understanding of what could take place when users have access to a service like Mobile Buddies. The scenarios help us understand the different modes of use and thus we are able to tailor content and functionality to the best possible user experience. Scenarios force the creators to think about tasks and goals. Furthermore, they contribute to keeping a strategic and long-term vision for the project.

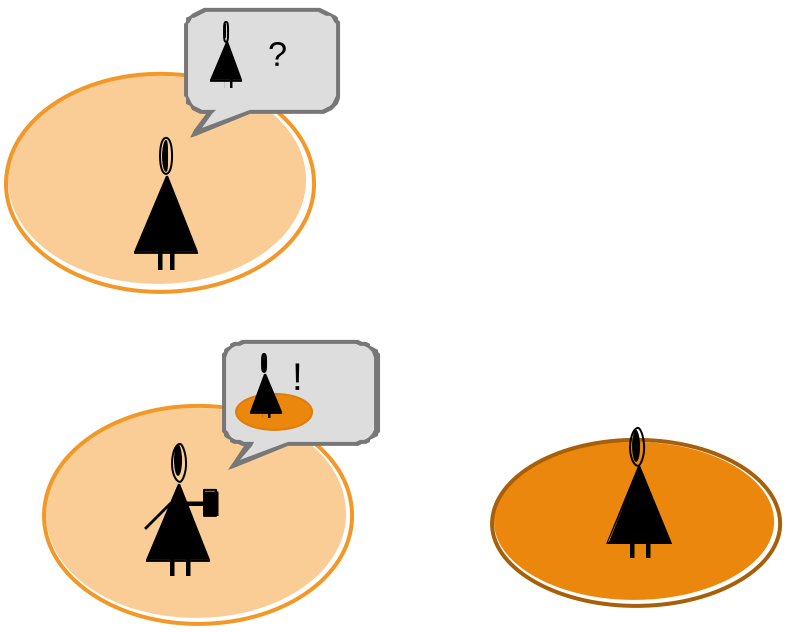
Below are a few scenarios illustrating how people could use Mobile Buddies in a future context, surrounding them with the trappings of their future lives. It is also important to bear in mind that new products and services create new behaviors.

### Find Bookmark



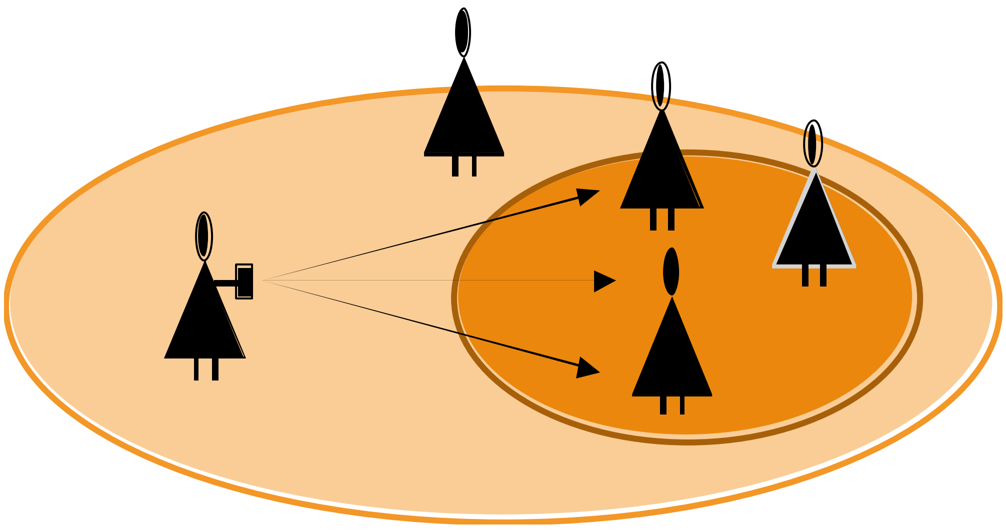
Maria is strolling around the East Village on a beautiful Sunday afternoon. Since she feels like having a cup of coffee, she logs on to Mobile Buddies from her mobile phone in order to find a nice café. She chooses to see bookmarks in the “current” area and she is presented with a number of cafes that her friends have recommended.

### Search Buddy



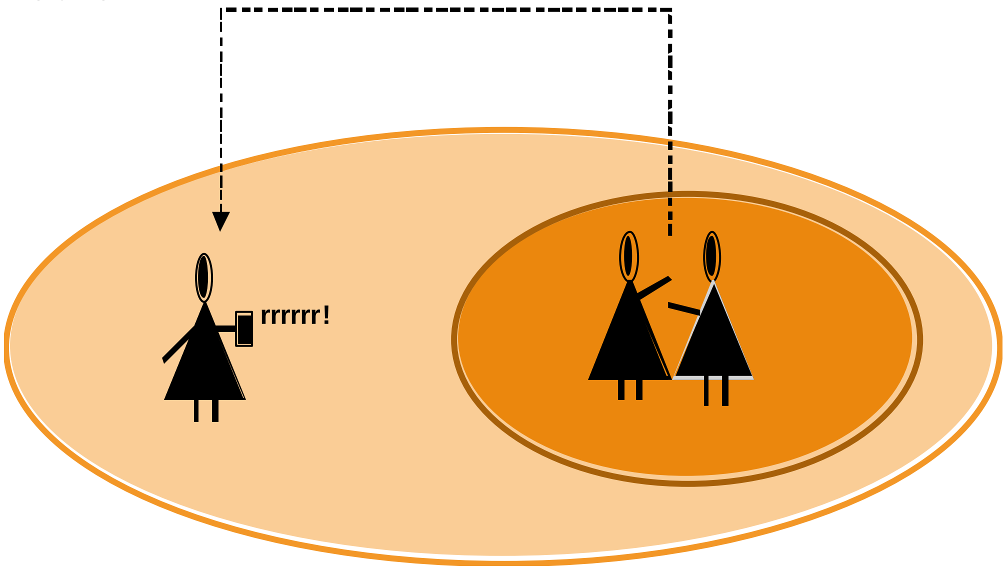
5.30pm; Sean is home cooking dinner for his 10 year old son Alex. When it is 6.30 pm Alex has still not come home. Supper is cold, there is no answer on Alex phone, and Sean is worried… He logs on to Mobile Buddies (web, SMS or WAP) and searches for Alex. After a few seconds he can feel relaxed again. Alex is at his best friends house...

### Search Buddies in area and send SMS



Sandy has just left work and is on her way home to Chelsea. She realizes she hasn't eaten since lunch and really wants to have a bite somewhere close to her home. She logs on to Mobile Buddies to find out what friends are in the area and sends them a message...

**Alert Me!**



Steve is out having a good time on a Friday night. Although the beer is good and the atmosphere is great he misses some of his friends, he wonder where they might be. In the next moment he feels his phone vibrate in his pocket. He picks it up and sees that Marcus and Josh are at Niagara. “Wow, I better leave for East 7th street now….”

## Information Architecture

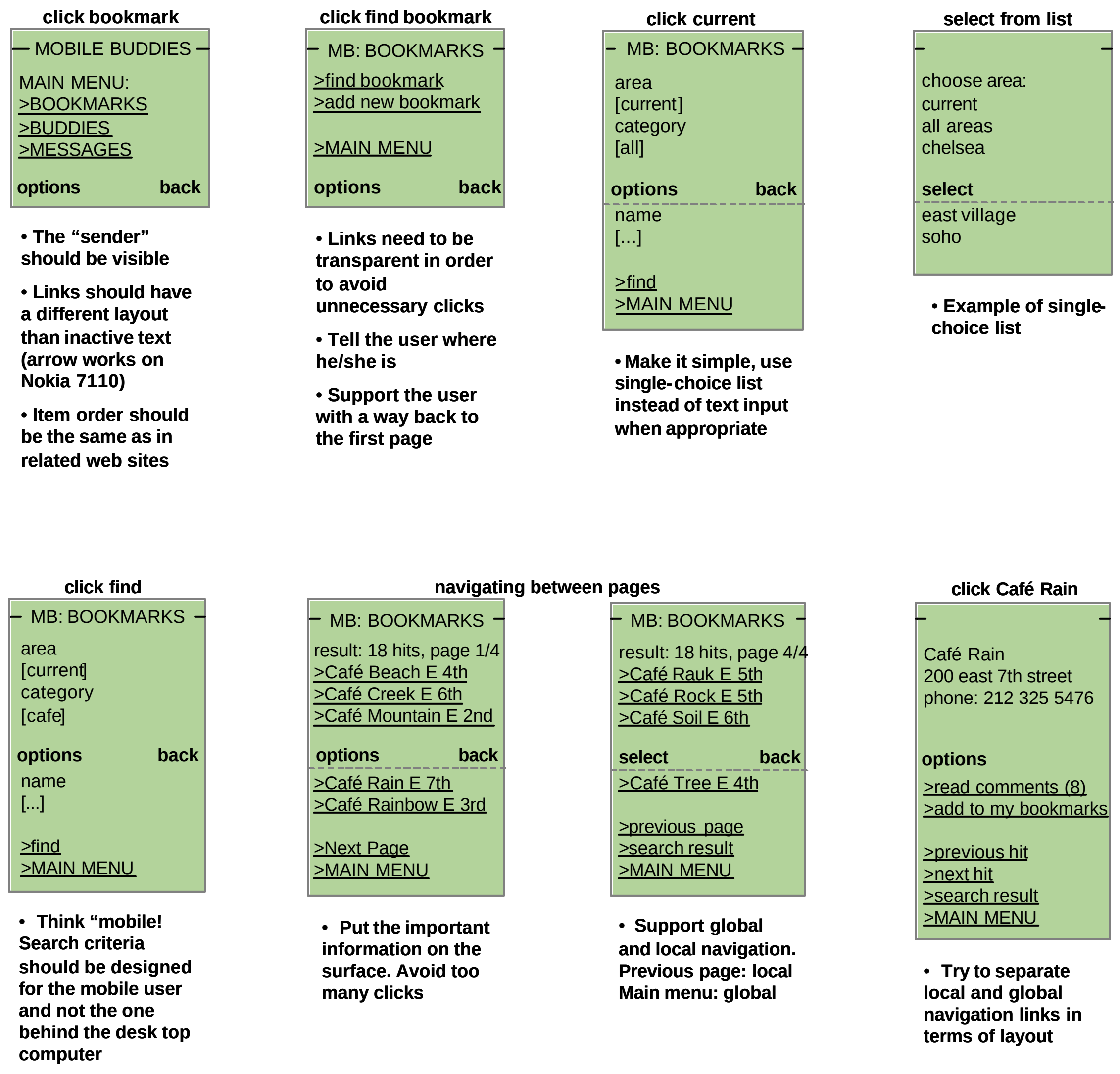
When using a phone for other reasons than those most users are familiar with, like calling and sending messages, it should be clear what functions are available at a certain moment and the result of an accomplished action should be easy to interpret. In other words, the interface design should ensure that the user understands what she or he can do within the service and that she is aware of what is happening. Having the limitations of small screens in mind, as they are finite in terms of showing context, menus, and visualization of alternatives, this way of thinking contributes to developing better and more successful products.

Using a phone to surf the Internet means taking a huge leap away from the fixation with looks, as found within web and user-centered design. That is why well-created information flow is even more important within mobile phone services.

The information architect should consider the mental model of the user, and since not the common people are familiar with mobile services today, one has to assume that the general user’s model of the system is pretty vague. Therefore, make services as “light” and easy as possible.

**Information architecture in Mobile Buddies**

The information flow beneath shows how a user tries to find a recommended café in the neighborhood where he or she is at the present time.



**click bookmark**

— MOBILE BUDDIES —
MAIN MENU:
>BOOKMARKS
>BUDDIES
>MESSAGES
**options** **back**

- The "sender" should be visible
- Links should have a different layout than inactive text (arrow works on Nokia 7110)
- Item order should be the same as in related web sites

**click find bookmark**

– MB: BOOKMARKS –
>find bookmark
>add new bookmark
>MAIN MENU
**options** **back**

- Links need to be transparent in order to avoid unnecessary clicks
- Tell the user where he/she is
- Support the user with a way back to the first page

**click current**

– MB: BOOKMARKS –
area
[current]
category
[all]
**options** **back**
name
[...]
>find
>MAIN MENU

- Make it simple, use single-choice list instead of text input when appropriate

**select from list**

choose area:
current
all areas
chelsea
**select**
east village
soho

- Example of single-choice list

**click find**

– MB: BOOKMARKS –
area
[current]
category
[cafe]
**options** **back**
name
[...]
>find
>MAIN MENU

- Think "mobile! Search criteria should be designed for the mobile user and not the one behind the desk top computer

**navigating between pages**

– MB: BOOKMARKS –
result: 18 hits, page 1/4
>Café Beach E 4th
>Café Creek E 6th
>Café Mountain E 2nd
**options** **back**
>Café Rain E 7th
>Café Rainbow E 3rd
>Next Page
>MAIN MENU

- Put the important information on the surface. Avoid too many clicks

– MB: BOOKMARKS –
result: 18 hits, page 4/4
>Café Rauk E 5th
>Café Rock E 5th
>Café Soil E 6th
**select** **back**
>Café Tree E 4th
>previous page
>search result
>MAIN MENU

- Support global and local navigation. Previous page: local Main menu: global

**click Café Rain**

Café Rain
200 east 7th street
phone: 212 325 5476
**options**
>read comments (8)
>add to my bookmarks
>previous hit
>next hit
>search result
>MAIN MENU

- Try to separate local and global navigation links in terms of layout

## Testing

As we mentioned earlier we view usability together with usefulness as a qualitative focus that should be taken from the very beginning in a user-centered design process. To withhold the usability and usefulness focus, testing and evaluating are very important activities that need to be carried out continuously throughout the whole design process to derive at the best solutions. The overall purpose of testing is to identify potential problems early on and to verify that the solution is truly user-centered and in keeping with the defined concept, user experience, brand, visual design and tonality.

A number of tests were performed during the development of Mobile Buddies, but those will not be discussed in the scope of this paper.

## Conclusion

Mobile solutions should be as simple and clear as possible, so that communicating with another person anywhere, anytime is a natural process requiring little thought or effort.

It is becoming increasingly evident that the quality of user experience has a direct impact on the sustainability and prolonged lifecycle of mobile products. Both hardware and software solutions provide opportunities to effectively guide behavior for using mobile devices, and have the potential to make mobility seem natural. In the near future, as technology enables solutions to respond better to the habits of its' users, devices may incorporate features that play increasingly on serendipity, pushing unexpected but useful and interesting information onto its users. But keep in mind, this is due to a thorough understanding of the users and such an understanding is gained through user intelligence. Ultimately, providing mobile solutions and services that are smart and responsive to user needs will improve the culture of mobility and make mobile devices an indispensable extension of one's self.

November 2001
ISSN 1404-3203
http://www.it.uu.se/