Proceedings of the Second International Workshop on

The Interplay between Usability Evaluation and Software Development

I-USED 2009

Editors:

Silvia Abrahao
Kasper Hornbæk
Effie Lai-Chong Law
Jan Stage

Title: Proceedings of the Second International Workshop on the Interplay between Usability Evaluation and Software Development (I-USED 2009)

Editors: Silvia Abrahão, Kasper Hornbæk, Effie L-C Law and Jan Stage

Also appeared online as CEUR-WS.org/Vol-490 in
CEUR Workshop Proceedings (ISSN 1613-0073)
(http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/)

I-USED 2009

2nd International Workshop on the Interplay between Usability Evaluation and Software Development

Held at Interact 2009 in Uppsala, Sweden on August 24, 2009

Motivation

Software development is highly challenging. Despite many significant successes, several software development projects fail completely or produce software with serious limitations, including (1) lack of usefulness, i.e. the system does not adequately support the core tasks of the user, (2) unsuitable designs of user interactions and interfaces, (3) lack of productivity gains or even reduced productivity despite heavy investments in information technology (Gould & Lewis 1985, Strassman 1985, Brooks 1987, Matthiasen & Stage 1992, Nielsen 1993, Attewell 1994, Landauer 1995).

Broadly speaking, two approaches have been taken to address these limitations. The first approach is to employ evaluation activities in a software development project in order to determine and improve the usability of the software, i.e. the effectiveness, efficiency and satisfaction with which users achieve their goals (ISO 1998, Frøkjær et al. 2000). To help software developers' work with usability within this approach, more than 20 years of research in Human-Computer Interaction (HCI) has created and compared techniques for evaluating usability (Lewis 1982; Nielsen & Mack 1994).

The second approach is based on the significant advances in techniques and methodologies for user interface design that have been achieved in the last decades. In particular, researchers in user interface design have worked on improving the usefulness of information technology by focusing on a deeper understanding on how to extract and understand user needs. Their results today constitute the areas of participatory design and user-centered design (e.g. Greenbaum & Kyng 1991, Beyer & Holtzblatt 1998, Bødker, Kensing & Simonsen 2004).

In addition, the Software Engineering (SE) community has recognized that usability does not only affect the design of user interfaces but the software system development as a whole. In particular, efforts are focused on explaining the implications of usability for requirements gathering (Juristo et al., 2007), software architecture design (Bass, John & Kates 2001; Bass & John 2003), and the selection of software components (Perry & Wolf 1992).

However, the interplay between these two fields, and between the activities they advocate to be undertaken in software development, have been limited. Integrating usability evaluation at relevant points in software development (and in particular to the user interface design) with successful and to-the-point results has proved difficult. In addition, research in Human-Computer Interaction (HCI) and Software Engineering (SE) has been done mainly independently of each other with no in substantial exchange of results and sparse efforts to combine the techniques of the two approaches. Larry Constantine, a prominent software development researcher, and his colleagues express it this way: "Integrating usability into the software development process is not easy or obvious" (Juristo et al. 2001, p. 21).

Theme & Goals

The goal of this workshop is to bring together researchers and practitioners from the HCI and SE fields to determine the state-of-the-art in the interplay between usability evaluation and software development and to generate ideas for new and improved relations between these activities. The aim is to base the determination of the current state on empirical studies. Presentations of new ideas on how to improve the interplay between HCI & SE to the design of usable software systems should also be based on empirical studies. Within this focus, topics of discussion include, but are not limited to:

- Which artifacts of software development are useful as the basis for usability evaluations?
- How do the specific artifacts obtained during software development influence the techniques that are relevant for the usability evaluation?
- In which forms are the results of usability evaluations supplied back into software development (including the UI design)?
- What are the characteristics of usability evaluation results that are needed in software development?
- Do existing usability evaluation methods deliver the results that are needed in user interface design?
- How can usability evaluation be integrated more directly in user interface design?
- How can usability evaluation methods be applied in emerging techniques for user interface design?
- How can usability evaluation methods be integrated to novel approaches for software development (e.g., model-driven development, agile development).

Target audience

Participants are accepted on the basis of their submitted papers. We aim at 15 with a maximum of 20 participants. The intended audience is primarily software engineering and human-computer interaction researchers who are working with the theme. The workshop should also be relevant for practitioners who have experiences with and ideas for improving the interplay between HCI and SE.

Relevance to the Field

The main contribution is the determination of state-of-the-art and the identification of areas for improvement and further research. The HCI field includes a rich variety of techniques for either usability evaluation or user interface design. But there are very few methodological guidelines for the interplay between these key activities; and more important, there are few guidelines on how to properly integrate these two activities in a software development process.

Workshop Organizers

- Silvia Abrahao, Universidad Politécnica de Valencia, Spain
- Kasper Hornbæk, University of Copenhagen, Denmark
- Effie Lai-Chong Law, ETH Zürich, Switzerland and University of Leicester, United Kingdom
- Jan Stage, Aalborg University, Denmark

Program Committee

- Nigel Bevan, Professional Usability Services, United Kingdom
- Ann Blandford, University College of London, United Kingdom
- Cristina Cachero, Universidad de Alicante, Spain
- Maria Francesca Costabile, University of Bari, Italy
- Peter Forbrig, Universität Rostock, Germany
- Asbjørn Følstad, SINTEF, Norway
- Emilio Insfran, Universidad Politécnica de Valencia, Spain
- Maristella Matera, Politecnico di Milano, Italy
- Philippe Palanque, IRIT, France
- Fabio Paternò, ISTI-CNR, Italy
- Isidro Ramos, Universidad Politécnica de Valencia, Spain
- Martin Schmettow, Passau University, Germany

Other Reviewers

- Emanuel Montero, Universidad Politécnica de Valencia, Spain

Workshop Website

http://users.dsic.upv.es/workshops/i-used09/

Table of Contents

Criteria for selecting methods in user-centred design

Nigel Bevan
Professional Usability Services
12 King Edwards Gardens, London W3 9RG, UK
mail@nigelbevan.com
www.nigelbevan.com

ABSTRACT

The ISO TR 16982 technical report which provides guidance on the use of usability methods is being revised as ISO 9241-230. This paper describes the procedure currently being suggested for selecting user-centred design methods. The best practices in ISO PAS 18152 are prioritised based on the assessed benefits and risks, then the most appropriate methods to achieve the best practices are identified.

SELECTING USER-CENTRED DESIGN METHODS

Previous approaches to methods selection have focussed on the strengths and weaknesses of individual methods (e.g. [3]), and their cost benefits (e.g [1]). However the reason for using usability methods is to make specific contributions to user-centred design. As Wixon [6] says, "the goal is to produce, in the quickest time, a successful product that meets specifications with the fewest resources, while minimizing risk". "In the world of usability work on real products embedded in a corporate and business framework, we must focus on factors of success, such as how effectively the method introduces usability improvements into the product."

The approach suggested in this paper is to first identify the necessary user centred design activities, then select the most appropriate methods based on the design and organisational context.

The proposed steps needed to select user-centred methods for a project are:

1. Identify which categories of human-system (HS) best practice activities in Annex A can increase business benefits or reduce project risks.

   For any category of system development activity in column 1 of Annex A, the UCD professional can reference the best practice activities in column 2 (and read the explanation of them in ISO PAS 18152 if necessary). They can then use Annex C to help judge to what extent carrying out or not carrying out these activities will influence the final usability of the product, and hence result in potential business benefits from improved usability, or in project risks from inadequate usability [2].

2. For the selected categories of best practice activities choose the most appropriate methods:

   a) To what extent will each possible method listed in column 3 of Annex A achieve the best practices?

      NOTE This relies on the expertise of the UCD professional supported by the documentation of the methods, such as that being developed by the Usability Body of Knowledge [5].

   b) How cost effective is each possible method likely to be?

      The most cost-effective methods can be selected by using Annex B to identify the method types, and then taking account of the associated strengths, weakness and constraints of each method type (examples of which are given in Annex D):

      - Constraints: time, cost, skills available, access to stakeholders and other users (Tables 4, 5 and 8 from 16982).
      - The nature of the task: complexity, amount of training required, consequences of errors, time pressure (Table 6 from 16982).
      - The nature of the product: whether new, complexity (Table 7 from 16982).
      - Context of use: range of contexts, how well understood (Table 9, to be done).

The selection of appropriate methods can be carried out as part of project planning, and may also be reviewed prior to each system development activity.

As the development of ISO 9241-230 is in the early stages, feedback on this proposed approach s welcomed.

REFERENCES

[1] Bevan, N. (2005). Cost benefits framework and case studies. In: Bias, R.G. & Mayhew, D.J. (eds) (2005). Cost-Justifying Usability: An Update for the Internet Age. Morgan Kaufmann.

[2] Bevan, N. (2008) Reducing risk through Human Centred Design. Proceedings of I-USED 2008, Pisa, September 2008.

[3] ISO TR 16982 (2002). Usability methods supporting human-centred design

[4] ISO PAS 18152 (2003). A specification for the process assessment of human-system issues.

[5] UPA (2009) Usability Body of Knowledge. www.usabilitybok.org

[6] Wixon, D. (2003) Evaluating usability methods: why the current literature fails the practitioner. Interactions, 10 (4) pp. 28-34.

Annex A. Examples of methods that can be used to support HS best practices

Activity category	Best practices for risk mitigation	UCD methods and techniques
1. Envisioning opportunities	•Identify expected context of use of systems [forthcoming needs, trends and expectations]. •Analyze the system concept [to clarify objectives, their viability and risks].	-Future workshop -Preliminary field visit -Focus groups -Photo surveys -Simulations of future use environments -In-depth analysis of work and lifestyles
2. System scoping	•Describe the objectives which the user or user organization wants to achieve through use of the system.	-Participatory workshops -Field observations and ethnography -Consult stakeholders -Human factors analysis
	•Define the scope of the context of use for the system.	-Context of use analysis
3. Understanding needs a) Context of use	•Identify and analyze the roles of each group of stakeholders likely to be affected by the system. •Describe the characteristics of the users. •Describe the cultural environment/ organizational/ management regime. •Describe the characteristics of any equipment external to the system and the working environment. •Describe the location, workplace equipment and ambient conditions. •Decide the goals, behaviours and tasks of the organization that influence human resources •Present context and human resources options and constraints to the project stakeholders.	-Success critical stakeholder identification -Field Observations and ethnography -Participatory workshop -Work context analysis -Context of use analysis -Event data analysis -Participatory workshops -Contextual enquiry
b) Tasks	•Analyze the tasks and worksystem.	-Task analysis -Cognitive task analysis -Work context analysis
c) Usability needs	•Perform research into required system usability.	-Investigate required system usability -Usability benchmarking -Heuristic/expert evaluation
d) Design options	•Generate design options for each aspect of the system related to its use and its effect on stakeholders. •Produce user-centred solutions for each design option.	-Early prototyping & usability evaluation -Develop simulations -Parallel design (tiger testing)
4. Requirements a) Context requirements	•Analyze the implications of the context of use. •Present context of use issues to project stakeholders for use in the development or operation of the system.	-Define the intended context of use including boundaries
b) Infrastructure requirements	•Identify, specify and produce the infrastructure for the system. •Build required competencies into training and awareness programs. •Define the global numbers, skills and supporting equipment needed to achieve those tasks.	-Identify staffing requirements and any training or support needed to ensure that users achieve acceptable performance
c) User requirements	•Set and agree the expected behaviour and performance of the system with respect to the user. •Develop an explicit statement of the user requirements for the system. •Analyze the user requirements. •Generate and agree on measurable criteria for the system in its intended context of use.	-Scenarios -Personas -Storyboards -Establish performance and satisfaction goals for specific scenarios of use -Define detailed user interface requirements -Prioritize requirements (eg QFD)
5. Architecting solutions a) System architecting	•Generate design options for each aspect of the system related to its use and its effect on stakeholders. •Produce user-centred solutions for each design option. •Design for customization. •Develop simulation or trial implementation of key aspects of the system for the purposes of testing with users. •Distribute functions between the human, machine and organizational elements of the system best able to fulfil each function. •Develop a practical model of the user's work from the requirements, context of use, allocation of function and design constraints for the system. •Produce designs for the user-related elements of the system that take account of the user requirements, context of use and HF data. •Produce a description of how the system will be used.	-*Function allocation* -Generate design options -Develop prototypes -Develop simulations
b) Human elements	•Decide the goals, behaviours and tasks of the organization [that influence human resources] •Define the global numbers, skills and supporting equipment needed to achieve those tasks. •Identify current tasking/duty •Analyze gap between existing and future provision •Identify skill requirements for each role	-Work domain analysis -Task analysis -Participatory design -Workload assessment -Human performance model -Design for alertness -Plan staffing

	•Predict staff wastage between present and future. •Calculate the available staffing, taking account of working hours, attainable effort and non-availability factor •Identify and allocate the functions to be performed Functional decomposition and allocation of function. •Specify and produce job designs and competence/ skills required to be delivered •Calculate the required number of personnel. •Generate costed options for delivery of training and/or redeployment •Evolve options and constraints into an optimal [training] implementation plan (4.3.5) •Define how users will be re-allocated, dismissed, or transferred to other duties. •Predict staff wastage between present and future. •Calculate the available staffing, taking account of working hours, attainable effort and nonavailability factor. •Compare to define gap and communicate requirement to design of staffing solutions.	
c) Hardware elements	See a) System architecting.	-Prototyping and usability evaluation -Physical ergonomics -Participatory design
d) Software elements	See a) System architecting.	-User interface guidelines and standards -Prototyping and usability evaluation -Participatory design
6. Life-cycle planning a) Planning	•Develop a plan to achieve and maintain usability throughout the life of the system. •Identify the specialist skills required and plan how to provide them.	-Plan to achieve and maintain usability -Plan use of HSI data to mitigate risks
b) Risks	•Plan and manage use of HF data to mitigate risks related to HS issues. •Evaluate the current severity of emerging threats to system usability and other HS risks and the effectiveness of mitigation measures. •Take effective mitigation to address risks to system usability.	-HSI program risk analysis
c) User involvement	•Identify the HS issues and aspects of the system that require user input. •Define a strategy and plan for user involvement. •Select and use the most effective method to elicit user input. •Customize tools and methods as necessary for particular projects/stages. •Seek and exploit expert guidance and advice on HS issues.	-Identify HSI issues and aspects of the system requiring user input -Develop a plan for user involvement -Select and use the most effective methods -Customize tools and methods as necessary
d) Acquisition	•Take account of stakeholder and user issues in acquisition activities.	-Common Industry Format
e) Human resources	•Implement the HR strategy that gives the organisation a mechanism for implementing and recording lessons learnt •Enable and encourage people and teams to work together to deliver the organization's objectives. •Create capability to meet system requirements in the future (conduct succession planning) •Develop and trial training solution to representative users. •Deliver final training solutions to designated staff according to agreed timetable. •Provide means for user feedback [on human issues].	
7. Evaluation a) Risks	•Assess the health and well-being risks to the users of the system. •Assess the risks to the community and environment arising from human error in the use of the system. •Evaluate the current severity of emerging threats to system usability and other HS risks and the effectiveness of mitigation measures. •Assess the risks of not involving end users in each evaluation.	-Risk analysis (process and product)
b) Plan and execute	•Collect user input on the usability of the developing system. •Revise design and safety features using feedback from evaluations. •Plan the evaluation. •Identify and analyze the conditions under which a system is to be tested or otherwise evaluated. •Check that the system is fit for evaluation. •Carry out and analyze the evaluation according to the evaluation plan. •Understand and act on the results of the evaluation.	-Obtain user feedback on usability -Use models and simulation
c) Validation	•Test that the system meets the requirements of the users, the tasks and the environment, as defined in its specification. •Assess the extent to which usability criteria and other HS requirements are likely to be met by the proposed design.	-Compare with requirements -Common Industry Format for usability reports -Performance measurement
d) HSI knowledge	•Review the system for adherence to applicable human science knowledge, style guides, standards, guidelines, regulations and legislation.	
e) Staffing	•Decide how many people are needed to fulfill the strategy and what ranges of competence they need.	*HR*

	•Develop and trial training solution to representative users. •Conduct assessments of usability [relating to HR]. •Interpret the findings •Validate the data. •Check that the data are being used.	
8. Negotiating commitments a) business case	•Contribute to the business case for the system. •Include HS review and sign-off in all reviews and decisions	-Program risk analysis
b) requirements	•Analyze the user requirements. •Present these requirements to project stakeholders for use in the development and operation of the system. •Identify any staffing gap and communicate requirement to design of staffing solutions.	-Value-based practices and principles (identify success critical stakeholder requirements) -Common Industry Specification for Usability Requirements -Environment/organization assessment
9. Development and evolution	•Maintain contact with users and the client organization throughout the definition, development and introduction of a system. •Evolve options and constraints into an implementation strategy covering technical, integration, and planning and manning issues. •	-Risk analysis (process and product) -User feedback on usability -Use models and simulation -Guidelines: Common Industry Format for usability reports -Performance measurement
10. Monitoring and control	•Analyze feedback on the system during delivery and inform the organization of emerging issues. •Manage the life cycle plan to address HS issues. •Take effective mitigation to address risks to system usability. •Take account of user input and inform users. •Identify emerging HS issues. •Understand and act on the results of the evaluation. •Produce and promulgate a validated statement of staffing shortfall by number and range of competence.	-Organizational and environmental context analysis -Risk Analysis -User feedback -Work context analysis
11. Operations and retirement a) Operations	•Analyze feedback on the system during delivery and inform the organization of emerging issues. •Produce personnel strategy. •Review the system for adherence to applicable human science knowledge, style guides, standards, guidelines, regulations and legislation. •Deliver training and other forms of awareness-raising to users and support staff. •Assess the effect of change on the usability of the system. •Review the health and well-being risks to the users of the system. •Review the risks to the community and environment arising from human error in the use of the system. •Take action on issues arising from in-service assessment. •Perform research to refine and consolidate operation and support strategy for the system.	-Work context analysis -Organizational and environmental context analysis
b) Retirement	•Collect and analyze in-service reports to generate updates or lessons learnt for the next version of the system. •Identify risks and health and safety issues associated with removal from service and destruction of the system. •Define how users will be re-allocated, dismissed, or transferred to other duties. •Plan break-up of social structures. •Debriefing and retrospective analysis for replacement system.	
12. Organizational capability improvement a) HSI capability data collection, analysis, and improvement	•Identify and use the most suitable data formats for exchanging HF data. •Have a policy for HF data management. •Perform research to develop HF data as required. •Produce coherent data standards and formats. •Define rules for the management of data. •Develop and maintain adequate data search methods. •Feedback into future HR procurement, training and delivery strategies.	-Assess and improve HSI capability
b) Organizational skill/career and infrastructure development planning and execution	•Define usability as a competitive asset •Set usability, health and safety objectives for systems •Follow competitive situation in the market place •Develop user-centred infrastructure. •Relate HS issues to business benefits. •Establish and communicate a policy for human-centeredness. •Include HR and user-centred elements in support and control procedures. •Define and maintain HCD and HR infrastructure and resources. •Increase and maintain awareness of usability. •Develop or provide staff with suitable HS skills. •Take account of HS issues in financial management •Assess and improve HS capability in processes that affect usability, health and	-Develop and maintain HSI infrastructure and resources -Identify required HSI skills -Provide staff with HSI skills -Establish and communicate a policy on HSI -Maintain an awareness of usability

	safety. •Develop a common terminology for HS issues with the organization. •Facilitate personal and technical interactions related to HS issues. •Feedback into future HR procurement, training and delivery strategies. •Create capability to meet system requirements in the future (conduct succession planning) •Identify any opportunities for redeployment. •Develop a strategy for [HR] data gathering	

ANNEX B. METHOD TYPES

This table (in its final version) will cross-reference the methods and techniques in Annex A to the different method types in the columns of the tables in Annex D, thus enabling the criteria in Annex D to be applied to the methods and techniques in Annex A.

	Method type											
Usability methods and techniques	**Observation of users**	**Performance-related measurements**	**Critical-incident analysis**	**Questionnaires**	**Interviews**	**Thinking aloud**	**Collaborative design and evaluation**	**Creativity methods**	**Document-based methods**	**Model-based methods**	**Expert evaluation**	**Automated evaluation**
Assess and improve HSI capability												
Cognitive task analysis												
Common Industry Format for usability reports												
Common Industry Specification for Usability Requirements												
Compare with requirements												
Consult stakeholders												
Context of use analysis												
Contextual enquiry												
Customize tools and methods as necessary												
Define detailed user interface requirements												
Define the intended context of use including boundaries												
Design for alertness												
Develop a plan for user involvement												
Develop and maintain HSI infrastructure and resources												
Develop prototypes												
Develop simulations												
Early prototyping and usability evaluation												
Environment/organization assessment												
Establish and communicate a policy on HSI												
Establish performance and satisfaction goals for specific scenarios of use												
Event data analysis												
Field observations and ethnography												
Focus groups												
Function allocation												
Future workshop												
Generate design options												
Guidelines: Common Industry Format for usability reports												
Heuristic/expert evaluation												
HR												
HSI program risk analysis												
Human factors analysis												
Human performance model												

Identify HSI issues and aspects of the system requiring user input												
Identify required HSI skills												
Identify staffing requirements and any training or support needed to ensure that users achieve acceptable performance												
In-depth analysis of work and lifestyles												
Investigate required system usability												
Maintain an awareness of usability												
Obtain user feedback on usability												
Organizational and environmental context analysis												
Parallel design (tiger testing)												
Participatory design												
Participatory workshop												
Performance measurement												
Personas												
Photo surveys												
Physical ergonomics												
Plan staffing												
Plan to achieve and maintain usability												
Plan use of HSI data to mitigate risks												
Preliminary field visit												
Prioritize requirements (eg QFD)												
Program risk analysis												
Prototyping and usability evaluation												
Provide staff with HSI skills												
Risk analysis (process and product)												
Scenarios												
Select and use the most effective methods												
Simulations of future working environments												
Storyboards												
Success critical stakeholder identification												
Task analysis												
Usability benchmarking												
Use models and simulation												
User feedback												
User feedback on usability												
User interface guidelines and standards												
Value-based practices and principles (identify success critical stakeholder requirements)												
Work context analysis												
Workload assessment												

ANNEX C: BUSINESS BENEFITS AND PROJECT RISKS

Developing a product with increased usability can provide business benefits (Table C1, column 1). Conversely, developing a product with inadequate usability can risk not achieving stated project objectives (Table C1, column 2).

The ultimate goal of system development is to produce a system that satisfies the needs of its operational stakeholders (including users, operators, administrators, maintainers and the general public) within acceptable levels of the resources of its development stakeholders (including funders, acquirers, developers and suppliers). Operational stakeholders need a system that is effective, efficient and satisfying. Developing and delivering systems that satisfy all of these success-critical stakeholders usually requires managing a complex set of risks such as usage uncertainties, schedule uncertainties, supply issues, requirements changes, and uncertainties associated with technology maturity and technical design.

The additional expenditure needed for human centred activities is often difficult to justify because the budget holder for project development often may not personally gain from the potential business benefits such as increased sales or reduced whole life costs. Project managers may therefore be more influenced by the risks of not achieving stated project objectives. It is thus useful to understand both the potential cost benefits of usability and the associated risks when justifying resources for usability.

Table C1. Benefits and risks associated with usability

Business benefit	Risk
A. Reduced development costs	**A: Increased development costs to produce an acceptable system**
• Detecting and fixing usability problems early in the development process	• Not detecting and fixing usability problems early in the development process
• Reducing the cost of future redesign or radical change of the architecture to make future versions of the product more usable	• Increasing the cost of future redesign or radical change of the architecture to make future versions of the product more usable
• Reduced costs due to only necessary functionality	• Increased costs due to unnecessary functionality
• Reduced costs due to minimising documentation	• Increased costs due to additional documentation
• Reducing the risk of product failure	• Product fails
B: Web site usability: improved web sales	**B: Web site usability: poor web sales**
• Users more frequently find products that they want to purchase	• Users cannot find products that they want to purchase
• Users more easily find additional information (e.g. delivery, return and warranty information)	• Users cannot find additional information (e.g. delivery, return and warranty information)
• Satisfied users are more likely to make repeat purchases	• Dissatisfied users do not make repeat purchases
• Users trust the web site (with personal information and to operate correctly)	• Users do not trust the web site (with personal information and to operate correctly)
• Users recommend the web site to others	• Users do not recommend the web site to others
• Web site increases sales through other channels	• Web site fails to increase sales through other channels
• Reduced support costs	• Increased support costs
C: Product usability: improved product sales	**C: Product usability: poor product sales**
• Improve the competitive edge by marketing the products or services as easy to use	• Competitors gain advantage by marketing competitive products or services as easy to use
• Satisfied customers make repeat purchases or recommend the product to others	• Dissatisfied customers do not make repeat purchases or recommend the product to others
• Good ratings for usability in product reviews	• Poor ratings for usability in product reviews
• Improve the brand	• Brand damage
D: Improved productivity: benefits to purchasing organisation	**D: Poor productivity: risks to purchasing organisation**
• Faster learning and better retention of information	• Slower learning and poorer retention of information
• Reduced task time and increased productivity	• Increased task time and reduced productivity
• Reduced employee errors that have to be corrected later	• Increased employee errors that have to be corrected later
• Reduced employee errors that impact on the quality of service	• Increased employee errors that impact on the quality of service
• Reduced staff turnover as a result of higher satisfaction and motivation	• Increased staff turnover as a result of lower satisfaction and motivation
• Reduced time spent by other staff providing assistance when users encounter difficulties	• Increased time spent by other staff providing assistance when users encounter difficulties
E: Reduced support and maintenance costs	**E: Increased support and maintenance costs**
• Reduced support and help line costs	• Increased support and help line costs
• Reduced costs of training	• Increased costs of training
• Reduced maintenance costs	• Increased maintenance costs

ANNEX D. EXAMPLES OF CRITERIA FOR METHOD SELECTION, FROM ISO TR 16982

Legend	
++	Recommended;
+	Appropriate;
When the cell is empty	Neutral;
–	Not recommended;
NA	Not applicable (NA).

Table 4 — The constraints of the environment on the project

Project characteristics	Methods											
	Observation of users	Performance-related measurements	Critical incidents analysis	Questionnaires	Interviews	Thinking aloud	Collaborative design and evaluation	Creativity methods	Document-based methods	Model-based methods	Expert evaluation	Automated evaluation
Very tight time-scale		–	–	–		–	–		+	–	++	+
Cost/price control		–	–		–	–		–	++	–	+	
High quality level of the product to be delivered as the dominant requirement	++	++	+	++	++	+	+	+	+	+	+	+
Need for an early information/feed-back/diagnosis	+			+	++		+	+			+	
highly evolving specifications	+	+	+	+	+	+	++	+				

Table 5 — Methods related to the user characteristics

User characteristics	Methods											
	Observation of users	Performance-related measurements	Critical incidents analysis	Questionnaires	Interviews	Thinking aloud	Collaborative design and evaluation	Creativity methods	Document-based methods	Model-based methods	Expert evaluation	Automated evaluation
Cannot be involved/accessed	NA	NA	NA	NA	NA	NA	NA	NA	+	+	+	+
Can be involved/accessed	++	++	+	++	++	+	++	+	+	+	+	+
Have a significant disability	++	+	+	+	++	+	++	+	+	–	+	–

The Usability Paradox

Mark Santcroos
*Communication, Media and Information Technology
Rotterdam University
P.O. Box 25035, 3001 HA
Rotterdam, The Netherlands
m.a.santcroos@hro.nl

Arnold Vermeeren
Human Information and Communication Design
Dept. of Industrial Design
Delft University of Technology
Landbergstraat 15, 2628 CE
Delft, The Netherlands
a.p.o.s.vermeeren@tudelft.nl

Ingrid Mulder*†
†ID-StudioLab
Dept. of Industrial Design
Delft University of Technology
Landbergstraat 15, 2628 CE
Delft, The Netherlands
mulderi@acm.org

ABSTRACT
This position paper describes the issues surrounding teaching Human Centered Software Engineering (HCSE). We identify a lack of methods in software engineering practices in new media projects. To get a better understanding of that, we are currently conducting a study to identify the current practices in the new media industry. The Human Centered ICT Toolkit is discussed, what it achieved, and what is missing. Looking at problems from an educational viewpoint, we propose to create an integrated HCSE approach by improving the descriptions of methods and their relations giving richer meaning to students and improving understanding of HCSE.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

General Terms
Management, Documentation, Design, Human Factors, Standardization

Keywords
Human Centered, Software Engineering, SE, Education, CASE, UCD, HCI

1. INTRODUCTION
The Rotterdam University's School of Communication, Media and Information Technology (CMI) has a broad scope, with six bachelor studies ranging from (visual) Design to Technical Informatics. With this wide range of expertise, CMI established a research center on Human Centered Information and Communication Technology to conduct research in the field of human-computer interaction, intelligent environments and exploiting these technologies to understand human behavior and user experience as well as informing the design of innovative technology and interactive media.

1.1 Mediatechnology
One of the six studies within CMI is Mediatechnology, a software engineering degree primarily developing for the web and mobile. This field distinguishes itself from traditional software in the degree of user interaction that is involved. Consequently it has usability as a higher priority. Based on discussions we have learned that (almost) none of the organizations in this industry are using formalized methods and methodologies in their development process. In order to get more insight into this new field of software engineering, recently a study has been started. In this study we want to identify distinctive characteristics of media projects and what methods are being used for software evaluation and usability testing.

1.2 Research approach
All of our third year students have an internship in industry. During this period students are asked to report in detail about how their employer's software engineering process is arranged. In particular, they look at how clients and end-users are involved, what methods, techniques and models are used, how and when tests and evaluations are performed. Where possible, also information about whether the projects are finished within time and budget will be gathered.

1.3 Human Centered ICT Toolkit
One of the recent results of our group is the Human Centered ICT toolkit[3]. It offers an overview of methods and tools available for different phases (research, concept, design, development and implementation) in a project. Figure 1 shows the (iterative) phases. The higher goal was to guide Human Computer Interaction (HCI) and Software Engineering (SE) researchers and practitioners (i.e. the students of our six bachelor studies) and to enable them to develop a shared understanding of the overlapping fields of SE and HCI. It was constructed for the following goals:

- A guide to easily get an overview of user centered design and evaluation methods for interactive software and media;
- An overview of methods, tools and techniques available in literature and online resources; student's realistic projects).
- An overview of methods, tools and techniques learned and applied in CMI courses;

Figure 1: The Human Centered ICT Toolkit

- An overview of techniques and tools being employed in different IT- and media enhanced sectors (i.e. student's realistic projects).

The results from the described industry survey as well as the toolkit presenting the overview of the existing methods will be guiding the design and teaching of the curriculum of all six studies.

While the toolkit is a good start in listing the various methods, not all of the original goals have been reached. In the next section we touch upon possible solution scenarios for the identified issues.

2. USABILITY PARADOX

Essential in HCSE is usability testing. In recent years many methods for user-centered design have been developed and many tools are at hand. The design has even been standardized by ISO 13407[1]. The Usability Engineering Lifecycle approach [4] is a well-known example of implementing the user-centered design approach showing how the various elements of such an approach link to software engineering activities. It refers to a general development approach (rapid prototyping) as well as to the Object-Oriented Software Engineering approach OOSE [2]. However, we still see many suboptimal products and interfaces that do not meet usability quality standards. Therefore it can be said that usability practices are apparently not consequently applied. Reasons for this can very from business to technical arguments. In the remainder of this work we reflect upon some directions for improving usability practice adoption.

2.1 Information loss

We define the (human centered) software development process as the product of the five phases analysis, concept, design, development and implementation, as taken from the toolkit model. In an ideal world, people involved in this process are all around the same table, together with all the stakeholders of their product. The real world is far from the ideal situation. However, different phases are handled by different people not sitting around the same table. Handover from one phase to the other is most often done in writing and important information gets lost, resembling the Chinese Whisper game, getting worse with every step.

2.2 Integrated approach

Currently, the model (and implementation) of the toolkit is flat which makes it unscalable when it grows and therefore it's adoption difficult. While it was not claimed that the model is linear, it does not ensure handover of information between the various phases. To remove the loss of information between the phases it is needed that a dimension is added that describes the input and output of every method. By giving the description more descriptive information, it can be made more valuable to students. By adding this dimension to the model and the methods described in it, students might get a better understanding of the intrinsic purpose of the methods and can also better judge how the various methods can be used together.

2.3 CASE tool

To facilitate the proposed integrated approach we argue for the development of a CASE tool build on top of the toolkit. This will primarily be targeted for education. We realize that the use for the real world could be limited, as it has to fit with regular workflow. By using the CASE tool in software engineering processes undertaken by students, all kinds of data can be gathered. This information could be used for further studies regarding the process.

3. CONCLUSIONS

In this position paper we identify that there is a problem with the adoption of usability practices. We also experience a lack of standardization in the field of new media software development. Our conclusion is that these both facts contribute to a fuzzy teaching model for HCSE.

Future results from the study that we initiated to identify software practices in this new media industry will hopefully give more guidance for developing HCSE courses. More descriptive information enables the students to have a better understanding of the issues at hand.

We also believe that the final outcome of this research will be of benefit to the new media industry if we can educate the market through our students.

4. ACKNOWLEDGMENTS

We thank our colleagues for the discussions. In particular Bas Leurs for his leading role on the toolkit and Geert de Haan for his comments on this paper.

5. REFERENCES

[1] ISO 13407:1999. *Human-centred design processes for interactive systems.* ISO, Geneva, Switzerland.

[2] I. Jacobson. *Object-Oriented Software Engineering: A Use Case Driven Approach.* Addison Wesley, 1992.

[3] B. Leurs and I. Mulder. UCD education beyond the textbook: practicing a human-centered attitude. In *proceedings of CHI Nederland Conference*, 2009.

[4] D. Mayhew. *The usability engineering lifecycle: A practitioner's handbook for user interface design.* Morgan Kaufmann Publishers, 1999.

Use of Abstraction during User Interface Development

A Position paper

Ebba Þóra Hvannberg
University of Iceland
Hjardarhaga 2-6, 107 Reykjavik, Iceland
+354 525 4702
ebba@hi.is

ABSTRACT

This paper postulates a thesis claiming that abstraction is an essential part of communication during user interface development, that models are a way of expressing those abstractions and that user interface developers and software engineers need the same language for communication. Motivated by described myths and desired model characteristics stated in the literature, several counterarguments and arguments are given to the thesis, backed up with results from empirical research studies. The paper concludes with a plan of action to bring the thesis forward.

Categories and Subject Descriptors

H5.2 [**User Interfaces**]: *models*

General Terms

Human Factors

Keywords

Models, Abstraction, Development

1. INTRODUCTION

During software development, good communication within a development team and between a team and the stakeholders is essential. Many development lifecycle models have been suggested, and since participatory design, most if not all lifecycle models have emphasized inclusion of users. Recent agile models include two characteristics which involve users; writing some kind of user stories and letting the buyer of the product decide upon the next features in the product. Agile methods also stress that communication within teams are important, but they do discourage heavy documentation, processes or tools usage. Communication within a team is sometimes between different roles. The gap between software engineering and user interface development has been addressed to an extent in the literature and the conclusion is that whatever method is used the difficulties in communication between the software developers and usability specialists must be tackled [1]. We can all agree that communication is important, but how, what and why? Engineers have long communicated by means of mathematics, structural (architectures) and behavioural models (electrical engineers). They communicate about materials, structures of buildings, input and output of processes or systems. Computer scientists on the other hand express things with logic or computer programs. Because it seems so easy to change programs or write new ones, unlike concrete materials such as metal or cement, programmers think that modeling is not necessary, and in the race for fast products to market, they skip the preparation and planning and dive right into the implementation [2].

Because of inherent complexity of software, or maintenance, computer scientists tend to abstract from details for easier comprehension during development. Much of the effort of research in software engineering has been on how to communicate and articulate this abstraction. Early, this abstraction appeared as functions, with input and output as descriptions of change of states of the machine, then as user defined data structures. This was followed by entity-relationship models which had a strong influence on data modelling. Finally, since few decades, abstraction has been dominant in object-orientation, where abstraction occurs in forms of inheritance and encapsulation. Reuse was the anticipated return on investment of abstraction, initially with the concept of classes but when that did not meet expectations, recent developments have centered more on components and services as a form of abstraction.

There have been different suggestions of specializing descriptions of user interfaces from descriptions of software in general, such as patterns for user interfaces, frameworks and classes to user interface programming [3]. Instead of specialization from general software models, others have designed models for user interfaces independent of software development, such as cognitive models [4] . With the advent of the web as a platform, specific languages have been developed and engineering tools developed such as Web-ML and OO-H [5]. There have thus been countless attempts to devise specific models for user interaction, and while they are useful as such they will probably not be widely used by software developers, or in an interdisciplinary team of software developers and user interface designers [1]. Those models which are based on software engineering models such as UML-Web based Engineering (UWE) are perhaps more likely to be used in such teams [6].

From time to time, or should I say continuously, the research community has been trying to discover why developers refrain from using models as abstractions of software, be it informal or formal. As in the case of formal methods, scientists have tried to elicit myths and refute them [7]. Sometimes these myths have been denied with facts, sometimes with general arguments. Myths are drawn from speculations, here say or common knowledge in the area. While it is useful to gather such tacit knowledge present it explicitly, we need to conduct more empirical research investigating usage of models. A few such studies and my own results have motivated several statements of arguments for user interface modeling. The next section states a thesis, followed by counter arguments and arguments for that thesis. The paper concludes with a plan of actions.

2. THE THESIS

I postulate that a good way to communicate user interface designs, and hence results of usability evaluations, is through abstractions. I postulate that models as a form of abstraction are the best way to discuss and argue about a user interface development. The models are created during the requirements, design activities, used to assist during evaluation, used to express, understand and even predict results of an evaluation, and used to redesign a user interface to fix a usability problem uncovered during evaluation. Finally, I claim that for the task we need **software development models** in order to bridge the gap between user interface and software designers.

3. COUNTER ARGUMENT

Probably, not everyone agrees with the stated position. In the following, let us examine some of the counter arguments, some of which are motivated by the characteristics of good models, i.e. *abstraction, understandability, accuracy, predictability and inexpensiveness* [2], others which are motivated by myths stated in the literature [1, 7].

3.1 Working software over comprehensive documentation

One of the principles of agile development is working software over comprehensive documentation. Daily face to face meetings and frequent changing of roles or activities is meant to make up for lack of documentation. Knowledge is tacit and not explicit [8].

Modelling is a process which aids in describing and abstraction what is to be built. In support of this counterargument Ambler [9] refers to Constantine [10] who says that it is a misconception that agilists do not model. The truth is, Ambler states, that they do model, but that they discourage extensive modelling up-front but encourage active modelling along the way. Ambler supports this by referring to agile methods' literature, but also acknowledges that the models are sometimes rather informal.

Further, some of the misunderstanding of models is that their impact is to be mainly achieved through the end product. Instead, modelling is a dynamic activity and much is gained by the interaction which the modelling activity facilitates

3.2 Models are difficult to create and few know how to make them

Not many modeling languages have been created exclusively for user interface design or for that matter software development. The predominant one for software development is UML and it is quite large containing a number of different model types. The types of problems architects describe are scattered information among different model views, incompleteness of models, disproportion, i.e. more details in some parts than others and inconsistencies between teams. Furthermore, architects claim that models are sometimes used informally and there are a lack of modeling conventions [11]. A study on the use of UML demonstrated that those with more UML experience used it more extensively than those with less experience, suggesting that analysts need time to learn how to use the UML language well [12].

While I agree that modeling can be an intricate activity, I don't think it is the models themselves that are difficult to create, but it is the activity of abstraction which is hard. Successful user interface designers will always need to learn how to abstract. Some will learn it through modeling; others will learn it implicitly as they gain experience. With models they are forced to do it but they can avoid it they don't use models, with unpredictable results.

3.3 Creating models are costly and not worth the effort

Creating models, especially if supporting tools are unavailable, can be a difficult and time consuming effort. Not only are models difficult to create but also evolve ensuring that the models are synchronized with the implementation. A survey says that 52.5 percent of practitioners finish modeling when the model is complete, 33.8 percent of practitioners say that a model is done when it has passed a review or an inspection, and 32.8 percent of practitioners say that the deadline is the stopping criterion. Whereas the completeness of a model is more often a stopping criterion in larger projects, a deadline is more often a halting criterion for smaller projects [11]. These numbers tell us that models are created in different ways, and in the cases where the models are not complete, developers do not take full advantage of the benefits of models, namely model driven development where code is automatically generated from models [2].

A study we conducted recently showed that over 30% of the defects could be blamed on faulty dialogue or navigational design, yet only a few of those defects were fixed [13]. Why? We speculate that the reason may be that it was estimated too difficult to fix the usability problems because the solutions required a revised user interface architecture and hence were too costly or even too difficult to make.

Our conclusion, from our own and other research studies, is that it is very costly not to create models, and that unless models are complete, their full benefits are not reaped.

3.4 Models are limited to describing those characteristics of user interfaces which do not concern presentation

Models, especially very abstract ones, do not capture experience very well. To understand emotional experience, we need a detailed contextual implementation.

A survey among 171 analysts showed that of seven different types of UML diagrams and narratives, class diagrams were used most frequently, with 73% of the analysts saying that they were used in at least two-thirds of their projects. Use case diagrams were second, use case narratives fourth (44%), but statechart diagrams came sixth, with less than 30% of the analysts saying that statecharts are used in at least 2/3 of the projects. On the other hand when analysts were asked to mark those diagrams which were never used, class diagrams ranked the lowest with only 3% to 25% for collaboration diagrams, ranked the highest [11].

In this same survey, respondents were asked for the usefulness of the different diagrams. Interestingly, whereas statechart diagrams were used much less frequently than class diagrams, they ranked second in usefulness after class diagrams.

If we were to ask user interface developers, I speculate that class diagrams are only useful for conceptual modelling, but activity diagrams and then state charts diagrams would be ranked higher in terms of providing new information not found in use case narratives.

Conceptual modelling is still very useful in user interface design. Our study showed that around 23% of defects uncovered could be attributed to wrong conceptual models [13]. As we see in UML there are a number of different types of diagrams and this is what we should aim for in user interface modelling, but we need to link the different models together such as the presentation models to the structural and behavioural models, or else the developers will complain that there is a disconnect between the models.

3.5 Users do not understand models

In a user-centered development, it is imperative to involve users at all stages of development. It is also critical to include a multi-disciplinary group of experts. Therefore, the communication language needs to be familiar to all. Undeniably, artifacts such as low-fidelity prototypes, story boards, scenarios and use case narratives are very accessible to users, and countless research papers have claimed the usefulness of informal models of user interaction design such as scenarios and prototypes.

The results of a study on how UML is used, partly refutes this counterclaim. While the study' results reveal that stakeholders are most likely to use use case narratives and use case diagrams, clients are involved in developing, reviewing and approving other components more than expected. All of the clients interviewed in the study welcomed the use of UML and some even showed insight into its usage [12]. As expected, client involvement is highest with use case narratives, 76%, and lowest for statechart diagrams.

What is worrying is that models which are not useful with clients may be useful for programmers, thus creating a gap between the two groups.

4. ARGUMENT

In this section we restate our claims and support them.

4.1 Abstraction is key to communication

With abstraction we are able to discuss main interactions and principles in the software without burying it in too many details. Abstraction makes it easier to plan, verify and design. Abstraction allows us to present different views of the user interaction.

4.2 Models are a good way to communicate during user interface development

Sketches, scenarios or storyboards are all different types of models, since they describe the real end product but leave out some of its details. Diaper states that "HCI is an engineering discipline and therefore must model the real world that is assumed to exist, notwithstanding how poor and partial might be our models of it." [14]. Diaper emphasises the importance of task models since they describe a series of events or activities in time. He doesn't exclude other models but says that they play a lesser role. Seffah and Metzker acknowledge that task models are widely used in the user interface community but warn that they may describe functionality more than usability, thus not fulfilling the objectives of the user interface developer.

One of the desirable characteristics of models is that they should be predictive. Prediction does not only include foreseeing the behaviour of the user and the system through simulation, but also modelling of the development activity itself and not just the artefacts. With increased emphasis on approaches for the whole lifecycle, including maintenance, we need to include models for evaluations of user interfaces. Modelling evaluation results should help us predict whether a defect is likely to be fixed, whether an evaluator is likely to uncover defects, whether components are likely to be faulty etc.

4.3 Software development models can serve user interaction design and other components' designs

In communication between people a disagreement is often due to misunderstanding. We say that people don't speak the same language. To close the gap between software engineers and user interaction designers they need to speak the same language. Different dialects can be permissible but not different languages.

5. CONCLUSION

Current research gives evidence that user interface designers need better help in their work. The number of defects found and the increasing criticality of user interfaces demands that we continue searching for better ways to communicate and apply abstractions in interaction designs.

The counter arguments stated in this position paper are however real threats to this believe. I think these threats can be lessened with the following plan of action:

1. Develop a domain specific modelling language for user interface design which can be used by an interdisciplinary team of user interface designers and software developers.

2. Offer tutorials and develop body of knowledge for user interface modelling as an abstraction and communication activity.

6. REFERENCES

1. Seffah, A. and E. Metzker, *The obstacles and myths of usability and software engineering.* Commun. ACM, 2004. **47**(12): p. 71-76.
2. Selic, B., *The pragmatics of model-driven development.* Software, IEEE, 2003. **20**(5): p. 19-25.
3. Myers, B.A. and M.B. Rosson, *Survey on user interface programming*, in *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1992, ACM: Monterey, California, United States.
4. de Haan, G., G.C. van der Veer, and J.C. van Vliet, *Formal modelling techniques in human-computer interaction.* Acta Psychologica, 1991. **78**(1-3): p. 27-67.
5. Abrahão, S., et al. *A Model-Driven Measurement Procedure for Sizing Web Applications: Design, Automation and Validation* in *MoDELS* 2007: Springer-Verlag
6. Koch, N. and A. Kraus. *The expressive power of UML-based engineering.* . in *Second International Workshop on Web Oriented Software Techonlogy (CYTED)*. 2002.
7. Bowen, J.P. and M.G. Hinchey, *Seven more myths of formal methods.* Software, IEEE, 1995. **12**(4): p. 34-41.
8. Pikkarainen, M., et al., *The impact of agile practices on communication in software development.* Empirical Software Engineering, 2008. **13**(3): p. 303-337.
9. Ambler, S., *Tailoring Usability into Agile Software Development Projects*, in *Maturing Usability, quality in Software, Interaction and Value*, Effie Lai-Chong Law, Ebba Thora Hvannberg, and G. Cockton, Editors. 2008, Springer-verlag London. p. 75-95.
10. Constantine, L. *Process Agility and Software Usability: Toward Lightweight Usage-Centered Design. Accessed on April 25, 2006.* 2001 [cited 2009; Available from: www.foruse.com/articles/agiledesign.pdf
11. Lange, C.F.J., M.R.V. Chaudron, and J. Muskens, *In practice: UML software architecture and design description*, in *Software, IEEE*. 2006. p. 40-46.
12. Dobing, B. and J. Parsons, *How UML is used.* Commun. ACM, 2006. **49**(5): p. 109-113.
13. Law, E.L.-C., et al., *Impacts of Classification of Usability Problems (CUP) on System Redesign* in *Usability and User-Centered Design in Software Development: Case Studies and Real Life Applications*, Ann Blandford, et al., Editors. 2010, in review, IGI
14. Diaper, D., *The discipline of HCI.* Interacting with Computers, 1989. **1**(1): p. 3-5.

Adoption of Usability Engineering Methods: A Measurement-Based Strategy

Eduard Metzker
Vector Informatik, Stuggart Germany
eduard.metzker@gmx.net

Ahmed Seffah
EHL-LHR, Lausanne Switzerland
seffah.ahmed@ehl.ch

ABSTRACT
In the context of a software development organization, two strategies are possible for introducing and institutionalizing new usability engineering methods. The first one, expert-based institutionalization, require to resort to third party companies or experts that can, based its previous expertise, assist the team in selecting, implementing and institutionalizing usability engineering methods and tools. The second one, a measurement-based strategy, is based on empirical evidence for learning and assessing the appropriateness, usefulness of a usability engineering method. This paper proposed to combine these two approaches in a single process metrics support environment for selecting and institutionalizing usability engineering methods. The proposed approach has been validated via in a cross-organizational empirical study involving several software engineers from five mediums to large sized software development companies.

Keywords
Metrics, usability, usability engineering methods, institutionalization, adoption, software developments organization

1. INTRODUCTION

Within the scope of this research, by adoption we refer to the process and the related tools for selecting the appropriate new software development technology while assessing their suitability to the project needs and size as well as the capability of the personnel to use effectively and efficiently the new established technology. Adoption has been always a key challenge for software development organizations [28]. It was reported that the management staff commitment and the involvement of the personnel represent the top factors that impact on the success with a new technology when first introduced [27, 29, and 30].

However, despite management efforts made by organizations to render the transition more "user-friendly", the associated help and training material, although precise and perfectly describing the new method are often delivered in an esoteric and unreadable language. Another important factor is that organizations and managers are usually overly optimist in their employees' ability to quickly master a new technology. The reality is that understanding how to apply the technology is a long and arduous process.

Furthermore, there is little hard evidence backing up new technology to be adopted, and their costs and benefits are rarely understood [1]. Without this data, choosing a particular technology or methodology for a project at hand essentially is a random act with many consequences [2]. The findings from a very large survey made by Standish group, new technology is one of the top ten reasons for projects failure or success [27].

In order to support and effective adoption, a new metrics-based approach comprising a process model and a support environment are presented in this paper. A large case study was developed to assess the acceptance of the approach by development teams. The evaluation involved 44 professional software engineers from five medium to large-sized organizations. The evaluation method is discussed including context, method, subjects, procedure and results. Implications of the results on the design of metrics-based strategy are discussed for adopting new technology and assessing their acceptance by project teams. Based on the results, a set of guidelines is derived to optimize the acceptance of metrics exploitation approaches by project personnel.

2. THE PROPOSED METRICS SUPPORT ENVIRONMENT

The overall goal of the proposed approach, called Adoption-Centric Usability Engineering (ACUE), is to facilitate the adoption of UE methods by software engineering practitioners and thereby improve their integration into existing software development methodologies and practices. ACUE is designed to support project teams in institutionalizing this abstract knowledge about UE methods and to help them transfer this knowledge into their development processes. UE methods are perceived as integrated into an existing software development process when they are adopted by the project team, i.e. when they are accepted and performed by the project team.

ACUE exploits empirical data collected in different projects to yield stronger evidence on how the method works in a certain context. The data form an empirical base to guide the improvement of UE methods and to facilitate the informed selection and deployment of UE methods in future projects. If this approach is applied repeatedly in a number of projects over time, it leads to an incremental construction of a body of evidence to guide usability engineering method selection (Figure 1).

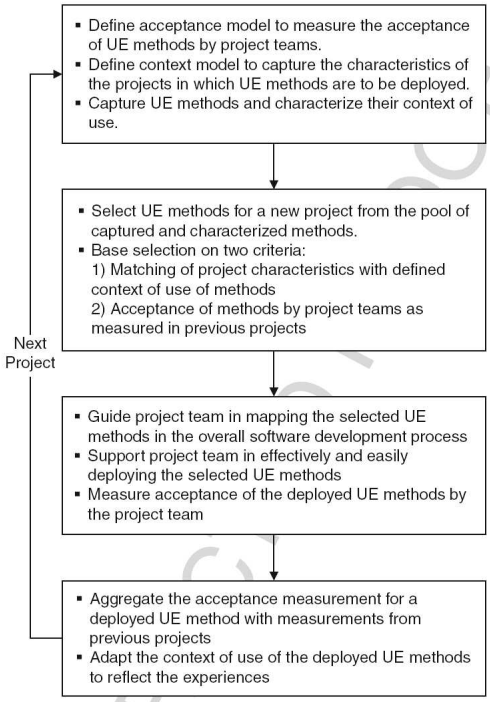

Figure 1: The overall view of ACUE approach

The support environment is called ESPrEE (Evidence-based Software PRocess Evolution Environment). The components of ESPrEE are integrated via a web portal and they be remotely accessed using any web browser. Its core functionalities are:

- At the beginning of a new project, the metrics-based method selection component of the environment is used to configure the set of usability engineering methods that will be applied in the project
- After the method selection has been completed, the environment generates a project-specific hyper-media workspace in which the methods selected are graphically visualized according to the project phases in which their usage is intended
- At the completion of major project phases or in post mortem sessions [13], the quality of the methods employed is assessed by the project team against a quality model. For this purpose quality models contain a set of quality factors and carefully defined rating scales

The core of the system is a fuzzy multi-criteria decision-making engine. The characteristics of the usability methods and projects are defined as sets of fuzzy sets. Based on these models the engine is able to compute similarity measures for projects and methods to facilitate decisions based on analogy. The engine is coupled with and assessment component. If a method receives a poor assessment in a certain project context, the method's characteristics are automatically adapted to reduce the probability of the method being selected in similar projects. On the other hand, if a method has successfully been applied in a certain project, its characteristics are adapted to increase its probability of selection in similar projects in the future.

The characteristics of the project are specified using the context models stored in the repository. A context model includes context factors to describe various project constraints such as the resources available in the project or the type of product to be developed. Each context model consists of a set of factors that can have nominal, ordinal or interval scale measures [11]. An example for an ordinal factor that describes a property of a product to be developed is 'user interface interaction complexity'. This factor may have three characteristics 'text-based interface', 'graphical user interface' or 'multi-modal interface'. Depending on the project characteristics, appropriate methods are suggested by the system. Candidate methods are ranked according to two different criteria: (1) similarity between the method and the project characteristics, (2) results of assessments from project teams that used the methods in previous projects.

Within, the system the usability engineering methods are provided in the format of method packages. Each package contains a textual description of a method that is structured according to the pyramid principle [12]. Auxiliary material such as templates, checklists and tools is linked to each package. This material facilitates easy compliance of the method described. The process guidance remains passive and does not enforce the performance of the methods proposed.

The constraints of industrial software development projects often enforce the invention of new methods or the adaptation and streamlining of existing methods. For these reasons the environments provides means for capturing methods and integrating them into the repository.

3. EVALUATION

3.1. Specific purpose of the evaluation

A large number of measurement programs are suspended or - in the worst case – failed. This is because the measurement program is not accepted by stakeholders for the following reasons [3, 4, 5, 6 and 7]:

1. The measurement process is perceived as tedious and time consuming
2. Effort and benefits of the program are poorly distributed
3. The impact on daily practice is perceived as being too low to justify sustained effort
4. Metrics support tools and/or processes are difficult to use

To examine if the proposed approach addresses these issues, the evaluation study was designed with the following questions in mind:

- Does each member of the project team understand the basic principles and structure of the method without extensive training?

- How do project managers assess the potential quantitative effects of the approach on their practices? Would they use the approach in their setting?
- Which problems the project personnel may face when applying the metrics support tool underlying the proposed framework?

3.2. Context of the evaluation

We used a set of five medium- to large-size software engineering companies developing advanced next-generation home entertainment systems, driver assistance technology for passenger cars and military support systems. The usability of these systems has been recognized by the organizations as a crucial quality factor. While usability engineering methods [10] are well-know by these companies to ensure the development of software with high usability, no experience with usability engineering was available in the engineering teams. ESPrEE was configured for this environment. Appropriate context and quality models were defined and usability engineering methods were captured in method packages. Resources included successful methods invented in previous industrial engineering projects such as reported in [16], methods distilled from literature on usability engineering [10, 17, 18], and recent research results such as Spencer's 'streamlined cognitive walkthrough'[19]. This initial population of the support tool was performed by a team of professional usability engineering experts and took about 2.5 man-months of effort. The participating organizations were part of a government-supported software engineering research consortium. However, no organization committed to adopt the approach prior to the evaluation.

3.3 The Subjects

All 44 subjects participated in the study on a voluntary basis. Of them, 39 are full-time employees working as software engineers for the companies described in section 3.2. Five subjects were graduate students working for the companies on a part-time basis. The subjects were involved in developing highly interactive software systems in a variety of domains, e.g. driver assistance systems, home entertainment, and military defense systems. Based on their experience with the development of highly interactive systems, the subjects were classified into three groups: new employees (NE, 10), software engineers (SE, 21), and usability engineers (UE, 13) [10]. In the following, these groups are referred to as user groups for reasons of simplicity.

3.4 Method

In this study, Davis' technology acceptance model (TAM) [14] was used. TAM's assessment dimensions 'perceived utility' and 'perceived ease of use' were extended while adding 'understandability' as a third dimension. TAM postulates that tool acceptance can be predicted by measuring two dimensions: the perceived usefulness (PU) and the perceived ease-of-use (PEU) of a system.

The perceived usefulness of the system expresses the "subjective probability that using a specific application system will increase (the user's) job performance within an organizational context", i.e. it is a measure for the perceived utility of the system.

The perceived ease-of-use is the main factor that influences the acceptance of a system. Davis defines perceived ease-of-use as "the degree to which the user expects the target system to be free of effort", i.e. it is a measure for the usability of a system.

Together, perceived ease-of-use and perceived usefulness constitute the person's attitude toward using a system. The attitude (A) and the perceived ease-of-use (PEU) influence the behavioral intention (BI) which can be used to predict the actual use of the system. The technology acceptance model (TAM)

TAM's dimension of perceived utility was further divided into the following sub-dimensions:

- Perceived compatibility with daily practice
- Perceived increase in efficiency
- Perceived usefulness
- Perceived support of working tasks

The sub-dimension of perceived utility was measured by qualitative effects analysis while usability was examined by studying user behavior during the user's interaction with the metrics support environment [10]. Understandability was examined via a knowledge test in which the subjects answered questions on the concepts of the overall approach and the metrics support environment. The knowledge test was performed before and after the interaction with the tool to study if and how the understanding of the concepts by the subjects changes.

To implement the study two basic techniques were deployed: questionnaires and scenario-based task solution. The purpose of the two questionnaires deployed (q1 and q2) are summarized in table 1.

Table 1: Purpose of the questionnaires deployed

	Data collected
q1	• Subject characteristics (age, qualification, professional experience) • Typical role of subject in the software engineering process • The subjects knowledge on the concepts of the approach and the metrics environment (pre-test, based on the introductory material supplied)

q2	• The subjects knowledge on the concepts of the approach and the metrics environment (post-test, based on the scenario-guided interaction with the metrics environment) • The subjects assessment of the utility and usability of the metrics environment

The scenario-based tasks that were specific for each user group forms the core of the evaluation. While the subjects were working on a task, behavioral data and any comments made while thinking out loud were captured as a basis for improving the metrics environment. Section 4.5 describes the tasks that were performed by the subjects while the exact evaluation procedure and deployment of the methods is described in section 4.6.

4.5 Tasks

To identify potential problems and study the acceptance of the metrics support environment under realistic conditions, specific usage scenarios were developed for each user group. Each scenario reflects the role of the subject and details the tasks to be solved.

All scenarios were embedded in a cover story that set a common background for all scenarios. Scenario S0 is equivalent for all user groups. In S0, the subjects were allowed to freely explore all components of the metrics environment. The other tasks to be solved in each scenario are different among the user groups (Table 2).

Table 2: Tasks to be solved in scenarios

	Tasks
NE-S1	Find an explanation of the term usability engineering. Mark the section for later exploration.
NE-S2	Find an introductory article about evaluation and tests. Review the material supplied.
NE-S3	Open the hypermedia workspace for the project DIGital. Find and open the method package on heuristic evaluation.
SE-S1	Browse all method packages available for project DIGital. Find and display a package on heuristic evaluation. Assess the quality of the method heuristic evaluation.
SE-S2	Comment the method ‚heuristic evaluation'. Edit the method ‚heuristic evaluation'. Extend the method package with a checklist to be used in ‚heuristic evaluation'.
SE-S3	Create a new method package. Fill the package with given raw input material. Specify the meta-data of the methods context model. Link the package to related packages.
UE-S1	Create a new project PORTAL. Choose a context model and specify the project characteristics via the context model. Choose appropriate method packages based on the project characteristics specified. Trigger generation of hypermedia workspace for the project PORTAL.
UE-S2	Manage access to the hypermedia workspace for the project PORTAL. Invite project team members. Manage access levels of project team members.

3.6. Test procedure and scripts

Prior to the evaluation sessions, all the subjects received introductory material. It described the concepts of the metrics approach and the components of the related environment. Single subjects, who, for some reason, had no access to the material prior to the evaluation, were given the opportunity to study printouts of the material. Each subject had an individual session, no group sessions were performed.

The evaluation session started with a short introduction of the participants, the procedure, and the objectives of the study. The subjects were explicitly informed that the goal of the evaluation was to assess the utility of the approach and not the capabilities of the participants and that all data would be treated confidentially. First questionnaire q1 was handed out. Next the tasks to be solved were handed out in form of scenarios. Scenario S0 was performed by each participant to promote a free exploration of the system. The time for S0 was limited to 20 minutes. Next, the group-specific scenarios were handed out to the subjects. No time limit was set for task completion. The subjects were encouraged to articulate impressions and problems and think aloud while performing the tasks. After the tasks of all scenarios were completed, questionnaire q2 was handed out to the subject. The evaluation is concluded with a brief free discussion.

Two observers were involved in each evaluation session. One observer recorded the behavioral data, while the other was responsible for writing down comments from the subjects were thinking aloud. During the session, the subjects worked with a laptop with each evaluation lasting of roughly two hours.

4. RESULTS AND FINDINGS

The qualitative effects analysis shows that the proposed metrics-based strategy is strongly accepted by the main target group. However the support environment receives higher-than-average ratings from all subject groups.

4.1 Understandability of the approach

The understanding of the metrics collection approach and the metrics support environment by the subjects was measured before and after usage of the support system via the

questionnaires q1 and q2. After reading the introductory material, the average percentage of correct answers was 36%. Subsequent to performing the scenario-based tasks, this value doubled, being up to 63%. The performance of the groups and the overall performance of the subjects are depicted in figure 2. It shows that even the relatively short time of usage of the metrics environment led to a significant increase in the understanding of the concepts and components of the approach. The increased understanding of the overall approach was lowest in the group of new employees (NE). However this can be easily explained since their scenarios (NE-S1-S3) did not comprise the usage of all components of the metrics support system.

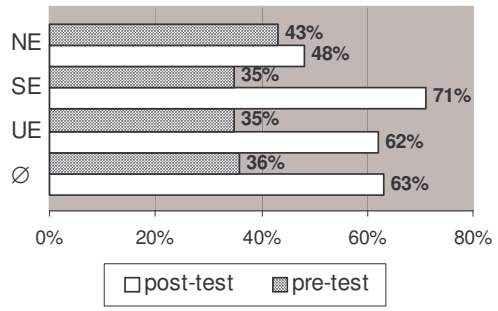

Figure 2: Pre- and post-test scores in knowledge tests with respect to subject groups (in percentage of correctly answered questions)

4.2 Usefulness of the approach

For the qualitative effects analysis, the subjects were asked to assess the properties of the metrics support environments along with the utility dimensions defined in section 3.4Each subject filled out questionnaire q2 after performing the scenario-specific tasks. For this reason questionnaire q2 includes a number of items to be rated on a five-level Likert scale [20] for each dimension. Figure 3 sets out the results of the qualitative effects analysis. The bars represent ratings of the assessment dimensions. The mean ratings were calculated for each dimension and grouped according to the subject groups.

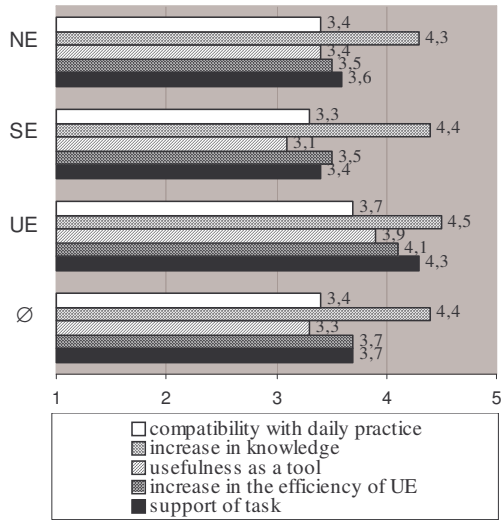

Figure 3: Results of the qualitative effects analysis (assessment scores for utility dimensions with respect to user group, 1: very low, 5: very high)

The results indicate that the approach and its support environment were generally assessed by all groups as higher-than-average useful and reasonable. All subjects seem to highly appreciate the potential of the approach for increasing their knowledge of usability engineering methods. The dimension 'task support' receives the highest scores from the UE group. This corresponds with the pre-configuration of the support environment with usability engineering methods and the subjects role as usability engineers. It could be concluded that the assessment of this dimension by the other groups could be further enhanced, if also usability engineering methods for other areas such as 'requirements engineering' or methods for enhancing programmer productivity were integrated into the method repository of the metrics environment. This result underpins the necessity to provide benefits for all groups of project personnel involved in metrics collection and exploitation.

4.3 Recommendations for usability improvements

The behavioral data and user comments recorded during task performance suggest that there is potential for improving the usability of the metrics support environment. The distribution of usability issues identified by subjects across the client components of the metrics support environment are presented in figure 4.

Most improvement suggestions are related to the components for process guidance and metrics-based decision support. The high number of usability issues identified for the process guidance component can be partially explained by the fact, that the scenarios of all user groups (NE, SE, UE) included interaction with the process guidance component.

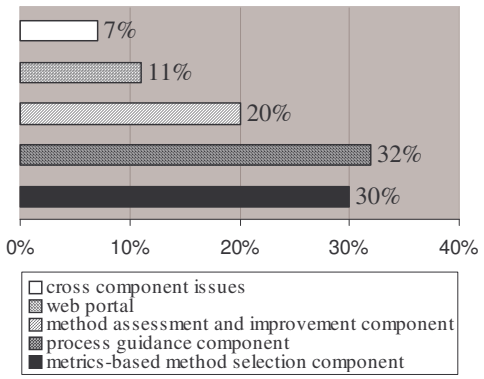

Figure 4: Distribution of the usability issues identified over client components of metrics support environment

One issue is that more assistance in working with the UE methods is appreciated by the subjects. In particular novice users could profit from concepts such as wizards to augment metrics capturing. Moreover the consistency between the applications should be increased. Finally the parts of the terminology used in the graphical user interfaces of the metrics environment should be revised for more comprehensible terms. One example given was that subjects suggested changing the term "project context model" to "project characteristics".

5. A CONCLUDING REMARK

In this paper, an approach for adopting UE methods was proposed. It consists of a three-phased process. First, usability engineering methods are selected for a new project based on the projects constraints. Then the project team is supported in deploying the methods in the project. Finally the project assesses the quality of the methods deployed. The approach is supported by a tool, an intranet that offers a web-based support system. The approach has been successfully implemented in industry.

Instead of evaluating the approach in an isolated longitudinal case study, a study was performed to examine the acceptance of the approach by practitioners from various organizations. The acceptance was measured in scenario-driven evaluation sessions, by capturing the understandability, perceived utility and perceived usability of the approach. The results indicate that the approach is accepted by the subject groups examined. We recommend using the study described as a template to estimate the initial acceptance when introducing tool supported measurement programs into organizations. Such studies can be useful for early identification of acceptance problems that hamper the log-term success of metrics programs. The usability of the metrics environment will be further improved using the feedback gathered.

6. ACKNOWLEDGEMENTS

Part of the empirical study presented in this paper was originally conducted at Daimler Chrysler. We would like to thank Elke Wetzstein and Gerd Jan Tschoepe from the Institute for Industrial Psychology of the Humboldt University Berlin for their support in preparing and conducting the evaluation sessions. Part of this research work was funded by the BMBF (German Ministry for Research and Education) under the project EMBASSI (01IL904I). We would like to thank also the National Science and Engineering Research Council of Canada and Daimler Chrysler for their financial support to the human-centered software engineering group at Concordia University.

REFERENCES

[1] D. E. Perry, A. A. Porter, and L. G. Votta, "Empirical Studies of Software Engineering: A Roadmap," in Proc. International Conference on Software Engineering (ICSE2000), 2000, pp. 345 - 355.

[2] V. R. Basili, F. Shull, and F. Lanubile, "Building Knowledge Through Families of Experiments," IEEE Transactions on Software Engineering, vol. 25, pp. 456-473, 1999.

[3] D. R. Goldenson, A. Gopal, and T. Mukhopadhyay, "Determinants of Success in Software Measurement Programs: Initial Results," in Proc. Sixth IEEE International Symposium on Software Metrics, 1999, pp. 10-21.

[4] B. G. Silverman, "Software Cost and Productivity Improvements: An Analogical View," IEEE Computer, vol. May 1985, pp. 86-96, 1985.

[5] R. T. Hughes, "Expert Judgment as an Estimating Method," Information and Software Technology, pp. 67-75, 1996.

[6] F. Niessink and H. Van Vliet, "Measurements Should Generate Value, Rather than Data.," in Proc. Sixth IEEE International Symposium on Software Metrics, 1999, pp. 31-39.

[7] O. Laitenberger and H. M. Dreyer, "Evaluating the Usefulness and the Ease of Use of a Web-based Inspection Data Collection Tool," in Proc. Fifth IEEE International Symposium on Software Metrics, 1998.

[8] S. Komi-Sirviö, P. Parviainen, and J. Ronkainen, "Measurement Automation: Methodological Background and Practical Solutions - A Multiple Case Study," in Proc. 7th IEEE International Software Metrics Symposium, 2001, pp. 306-316.

[9] L. Rosenberg and L. Hyatt, "Developing a Successful Metrics Program," in Proc. 8th Annual Software Technology Conference, 1996.

[10] J. Nielsen, Usability Engineering: Morgan Kaufman Publishers, 1994.

[11] L. Briand, K. El Emam, and S. Morasca, “On the Application of Measurement Theory in Software Engineering,” Empirical Software Engineering, vol. 1, pp. 61-88, 1996.

[12] B. Minto, The Pyramid Principle - Logic in Writing and Thinking, 3rd ed. London: Minto International Inc., 1987.

[13] A. Birk, T. Dingsøyr, and T. Stålhane, “Postmortem: Never Leave a Project without it,” IEEE Software, vol. 19, pp. 43-45, 2002.

[14] F. D. Davis, “A Technology Acceptance Model for Empirically Testing New End-User Information Systems: Theory and Results,,” in MIT Sloan School of Management. Cambridge, MA, USA: MIT Sloan School of Management, 1986.

[15] B. A. Kitchenham, “Evaluating Software Engineering Methods and Tools,” ACM SIGSoft Software Engineering Notes, pp. 11-15, 1996.

[16] E. Metzker and M. Offergeld, “An Interdisciplinary Approach for Successfully Integrating Human-Centered Design Methods Into Development Processes Practiced by Industrial Software Development Organizations,” in Engineering for Human Computer Interaction: 8th IFIP International Conference, EHCI 2001(EHCI'01), Lecture Notes in Computer Science, R. Little and L. Nigay, Eds. Toronto, Canada: Springer, 2001, pp. 21-36.

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

[18] D. J. Mayhew, The Usability Engineering Lifecycle: A Practioner's Handbook for User Interface Design: Morgan Kaufman Publishers, 1999.

[19] R. Spencer, “The Streamlined Cognitive Walkthrough Method: Working Around Social Constraints Encountered in a Software Development Company,” in Proc. Conference on Human Factors in Computing Systems (CHI'00), 2000, pp. 353-359.

[20] C. M. Judd, E. R. Smith, and L. H. Kidder, Research Methods in Social Relations, 6 ed: Harcourt Brace Jovanovich College Publishers, 1991.

[21] G. F. Smith and S. T. March, “Design and Natural Science Research on Information Technology,” Decision Support Systems, vol. 15, pp. 251-266, 1995.

[22] R. D. Galliers and F. F. Land, “Choosing Appropriate Information Systems Research Methodologies,” Communications of the ACM, vol. 30, pp. 900-902, 1987.

[23] T. DeMarco and T. Lister, Peopleware: Productive Projects and Teams, 2. ed. New York: Dorset House Publishing, 1999.

[24] Y. Malhotra and D. F. Galletta, “Extending the Technology Acceptance Model to Account for Social Influence: Theoretical Bases and Empirical Validation,” in Proc. Thirty-Second Annual Hawaii International Conference on System Sciences, 1999, pp. pp. 1006.

[25] A. Cockburn, Agile Software Development: Addison Wesley Longman, 2001.

[26] N. Juristo and A. M. Moreno, Basics of Software Engineering Experimentation. Dordrecht: Kluwer Academic Publishers, 2001.

[27] Standish Group. “CHAOS Chronicles or CHAOS: A Recipe For Success”. 1995.

[28] Bayer, J. and Melone, N. Adoption of Software Engineering Innovations in Organizations. Technical Report CMU/SEI-89-TR-017, Software Engineering Institute, Carnegie Mellon University.

[29] Desmarais M.C., Leclair R., Fiset J.Y., Talbi H. “Cost-Justifying Electronic Performance Support Systems.” Communications of the ACM, Vol. 40, No. 7, July 1997.

[30] Howard R. “Software Process Maturity: Measuring Its Impact on Productivity and Quality.” IEEE International Software Metrics Symposium, May 1993.

Towards a Usability Evaluation Process for Model-Driven Web Development

Adrian Fernandez
ISSI Research Group
Department of Information Systems and Computation - Universidad Politécnica de Valencia, Camino de Vera, s/n, 46022, Valencia, Spain.
+34 96 387 73 50
afernandez@dsic.upv.es

Emilio Insfran
ISSI Research Group
Department of Information Systems and Computation - Universidad Politécnica de Valencia, Camino de Vera, s/n, 46022, Valencia, Spain.
+34 96 387 73 50
einsfran@dsic.upv.es

Silvia Abrahão
ISSI Research Group
Department of Information Systems and Computation - Universidad Politécnica de Valencia, Camino de Vera, s/n, 46022, Valencia, Spain.
+34 96 387 73 50
sabrahao@dsic.upv.es

ABSTRACT
This paper presents an approach to integrate usability evaluations into Model-Driven Web development processes. Our main motivation is to define a generic usability evaluation process which can be instantiated into any concrete Web development process that follows a Model-Driven Development (MDD) approach. A preliminary version of a Web Usability Model was defined in order to support this usability evaluation process at several stages. This Web Usability Model decomposes the usability sub-characteristics (from the Software Quality Model proposed in the ISO/IEC 25000 SQuaRE standard) into other sub-characteristics and measurable attributes. Web metrics are intended to be associated to measurable attributes in order to quantify them. Our approach is intended to perform usability evaluations at several stages of a Web Development process. In this paper, we show how usability evaluations at final user interface (UI) can provide feedback about changes in order to improve usability issues at intermediate artifacts (Platform-Independent Models and Platform-Specific Models) or at transformations rules among these intermediate artifacts.

Categories and Subject Descriptors
D.2.9 [**Management**]: *Software quality assurance*, D.2.8 [**Metrics**]: *product metrics*. H5.2 [**User Interfaces**]: *Evaluation/methodology*

General Terms
Measurement, Design.

Keywords
Web Usability Model, Usability Evaluation, Web Metrics, Model-Driven Development.

1. INTRODUCTION
Usability in Web applications is a crucial factor since the ease or difficulty that users experience with this kind of systems will determine their success or failure. Web applications are increasing its importance in industrial domains; thereby, the need for usability evaluation methods that are specifically crafted for the Web domain has become critical.

I-USED '09, August 24, 2009, Upssala, Sweeden.

Usability evaluations methods for Web applications can be supported by a quality model which defines usability as a quality characteristic that is decomposed into specific attributes that are easier to measure. Although there are several proposes in this field, most of these approaches [12],[13] only consider usability evaluation at final stages when the product is almost completed where correcting its usability problems is more difficult. It is widely accepted that evaluations performed at each phase of Web applications development is a critical part of ensuring that the product will actually be used and be effective for its intended purpose. We argue that integrating usability issues into the MDD approach can be an effective way to reach this objective since the quality evaluation of intermediate artifacts (models that specify an entire Web application), is applied in all steps of the process [2]. A Web development process that follows a MDD approach basically transforms models that are independent from implementation details (Platform-Independent Models - PIM) into other models that contain specific aspects from a concrete platform (Platform-Specific Models - PSM). Transformation rules, which are applied at PSMs, are able to automatically generate the Web application source code (Code Model - CM).

This paper presents an approach to integrate usability evaluation into any Model-Driven Web Development method by defining a usability evaluation process. This Web Usability Model has been defined by decomposing the usability sub-characteristics (from the Software Quality Model proposed in the ISO/IEC 25000 SQuaRE standard) into other sub-characteristics and measurable attributes taking into account ergonomic criteria proposed in Bastien and Scapin [4]. Although our approach is intended to perform usability evaluations at several stages of a Web development process, in this paper, we mainly focus on how evaluations at final user interface (Code Model) can provide feedback about changes in order to improve usability issues at intermediate artifacts (PIM and PSM models) produced at early stages of the Web development process and at transformations rules among these intermediate artifacts.

This paper is organized as follows. Section 2 discusses related work that report usability evaluation processes for Web applications. Section 3 presents our approach to integrate usability evaluations into Model-Driven Web Development. Section 4 presents our Web Usability Model that supports our approach. Section 5 shows a brief example of how the usability evaluation process can be instantiated into a concrete Web development method. We mainly focus on evaluations at final user interface. Finally, Section 6 presents discussions and further work.

2. RELATED WORK

There are several approaches that deal with Web usability evaluation, for instance, Ivory [16], Olsina and Rossi [13], Calero *et al.* [5], Seffah *et al.* [15], and Moraga *et al.* [12].

Ivory [16] presents a methodology for evaluating information-centric Web sites. The methodology proposes five stages: identifying an exhaustive set of quantitative interface measures such as the amount of text on a page, color usage, consistency, etc; computing measures for a large sample of rated interfaces; deriving statistical models from the measures and ratings; using the models to predict ratings for new interfaces; and validating model prediction.

Olsina and Rossi [13] proposed the Web Quality Evaluation Method (WebQEM) to define an evaluation process in four technical phases: Quality requirements definition and specification (specifying characteristics and attributes based on the ISO/IEC 9126-1 [9]. such as *usability*, *functionality*, *reliability,* and *effectiveness* and taking into account Web audience's needs), elementary evaluation (applying metrics to quantify attributes), global evaluation (selecting aggregation criteria and a scoring model), and conclusion (giving recommendations). Nevertheless, evaluations take place mainly when the application is completed.

Calero *et al.* [5] present the Web Quality Model (WQM), which is intended to evaluate a Web application according to three dimensions: Web features (*content*, *presentation*, and *navigation*); quality characteristics based on the ISO/IEC 9126-1 (*functionality*, *reliability*, *usability*, *efficiency*, *portability*, and *maintainability*); and lifecycle processes (*development*, *operation* and *maintenance*) including organizational processes such as *project management* and reuse *program management*. WQM has been used to classify, according to these three dimensions, a total of 326 Web metrics taken from the existing literature. An evaluation process can be defined by selecting the most useful set of metrics to construct a "total Web quality" expression that could be used to quantify the quality of a given Web application. However, guidelines on how to define this process have not been provided.

Seffah *et al.* [15] present the Quality in Use Integrated Measurement (QUIM) as a consolidated model for usability measurement in Web applications. An editor tool has presented to define measurement plans collecting data from different combinations of metrics proposed in the model. QUIM combines existing models from ISO/IEC 9126-1 [9], ISO/IEC 9241-11 [8], and others. It decomposes usability into factors, and then into criteria. In this approach, a criterion can belong to different factors. Finally, these criteria are decomposed into specific metrics that can quantify the criteria.

Moraga *et al.* [12] present a usability model towards portlet evaluation. Portlets are pluggable user interface software components that are managed and displayed in a web portal. The portlet usability model is based on the sub-characteristics from ISO/IEC 9126 (understandability, learnability and compliance), nevertheless, the operability sub-characteristic was replaced by customizability which is closer to the portlet context. The usability evaluation process proposed is based on a number of ranking with acceptance thresholds in order to quantify the sub-characteristics from the models.

The majority of these approaches evaluate Web applications in order to suggest changes at design or implementation stages. It implies that more efforts and resources must be invested into code maintenance. This fact does not occur in a MDD approach where only the maintenance of models is required since source code can be automatically generated from the intermediate artifacts (PIM and PSM models).

In previous work, Abrahão and Insfran [3] proposed a usability model for early evaluation in model-driven architecture environments. Usability was decomposed into the same sub-characteristics as the ones in the ISO/IEC 9126 (*learnability*, *understandability*, *operability*, and *compliance*), and then decomposed again, into more detailed sub-characteristics and attributes. However, the model did not provide metrics for measuring the model attributes and it was not proposed specifically for the Web domain. Panach *et al.* [14] presents an adaptation from the previous model to the Web domain in order to evaluate usability at PIM models for a concrete and proprietary Model-Driven Web Development approach.

As far as we know, there is no proposal for a generic usability evaluation process supported by a usability model in the Model-Driven Web Development context.

3. THE USABILITY EVALUATION PROCESS

Since the adoption of Model-Driven Development (MDD) in the industrial domain has increased recently, our approach is intended to integrate usability issues into a Model-Driven Web Development. Web development methods that follow this approach such as OO-H [7], WebML [6], or UWE [11] support the development of a Web application by defining different views (models), including at least one structural model, a navigational model, and an abstract presentation model. These methods also provide model transformations and automatic code generation.

The usability of a Web application obtained as a result of a MDD process can be assessed at different abstraction levels (PIM, PSM and CM). Our proposal is intended to use a Web Usability Model, which is a set of sub-characteristics decomposed into measurable attributes that can be quantified by metrics. The Web Usability Model can be applied to assess the models from each abstraction level (see Fig.1). However, not all the measurable attributes can be evaluated at all the abstraction levels. The higher abstraction level, the less attributes can be considered. In addition, feedback that is obtained after the artifact evaluation has different targets depending on the abstraction level:

1. At the PIM level it is possible to assess models that specify the Web application independently of platform details such as: presentation models, navigational models, dialogue models, etc. (1 in Fig.1). The set of measurable attributes that can be evaluated at this level is mainly related to how the information will be accessed by users and how this information will be presented by abstract UI patterns (i.e. navigability, information density, etc). However, this set of attributes may differ depending on the PIM expressiveness from each Web development method. This evaluation will generate a usability report in order to provide feedback about how to correct these PIM models.
2. At the PSM level it is possible to assess the concrete interface models related to a specific platform (2 in Fig.1). The set of measurable attributes that can be evaluated at this level is wider since it includes attributes related with specific software components (widgets) that cannot be considered at PIM level (i.e. behavior of explore bars, visual feedback from radio buttons, etc).This evaluation will generate a usability report in

order to provide feedback to previous stages about how to correct the PIM and PSM models, as well as the transformation rules among them.

3. At the CM level it is possible to evaluate the final user interface (3 in Fig.1). The set of measurable attributes that can be evaluated at this level is the widest since more aspects related to the end-user perspective can be considered (i.e. browser compatibility, metaphor recognition, subjective appealing, etc). This evaluation will also generate a usability report in order to provide feedback to previous stages about how to correct the PIM and PSM models, as well as the transformation rules among them, and code generation rules among PSM and CM.

The former evaluations can be applied in an iterative way until the models (PIM, PSM, and CM) have the required level of usability. In order to integrate these evaluations into a framework, a usability evaluation process should be defined as an inspection method that guides evaluators on how the Web Usability Model can be applied. This inspection method could be defined in order to be compliant with the Quality Evaluation Division proposed in the ISO/IEC 2504n SQuaRE series [10]. The main steps to be included are:

1. Establish evaluation requirements such as the purpose of evaluation, identification of Web application type, and selection of the more relevant sub-characteristics of the Web Usability Model taking into account the users' needs.
2. Specify the evaluation concerning with the establishment of the artifacts to be evaluated (PIM, PSM or CM); selection of metrics associated to the attributes selected from the Web Usability Model; specification of the calculation formulas of these metrics taking into account the abstraction level of the artifact and the modeling primitives from the concrete Model-Driven Web development method; establishment of rating levels for these metrics; establishment of criteria for global assessment; and the definition of templates to report usability problems.
3. Design the evaluation plan describing the evaluator tasks schedule.
4. Execute the evaluation by applying the selected Web metrics in order to detect usability problems.
5. Generate the usability reports providing feedback in order to improve the intermediate artifacts (PIM and PSM) or transformation rules.
6. Analysis of changes suggested by usability reports and selection of the alternatives taking into account criteria such as level and priority of usability problems, resources needed to apply changes, etc.

It should be noted that this process is defined to be instantiated into any concrete Model-Driven Web Development method. The instantiation implies to know the modeling primitives of the concrete Model-Driven Web development method in order to be able to specify the calculation formula of the metrics and to understand the traceability between models. This traceability helps the evaluator to establish the source of the usability problems (PIMs, PSMs or transformations rules among them).

4. THE WEB USABILITY MODEL

The SQuaRE standard [10] proposes three different views for a quality model. These views are related to the context where the model will be applied: *Software Quality Model* to evaluate a concrete software product; *Data Quality Model* to evaluate the quality of the data managed in the product; and *Quality in Use Model* to evaluate how the stakeholders achieve their goals in a specific context of use.

Our Web Usability Model is an adaptation and extension from the usability model for model-driven development presented in Abrahão and Insfran [3], specifically, the model was adapted to be compliant with the *Software Quality Model* proposed in the SQuaRE.

The main quality characteristics of the software quality model are: *functionality*, *security*, *interoperability*, *reliability*, *operability (usability)* and *efficiency*. Although the term *operability* and *ease of use* have been proposed in SQuaRE to rename *usability* and *operability* sub-characteristic, respectively, we prefer to use the term *usability* and *operability* in this work to avoid misunderstandings in terminology.

Usability can be decomposed into the five sub-characteristics proposed in SQuaRE [10]: *learnability*, *understandability*, *ease of use (operability)*, *attractiveness* and *compliance*. The former three sub-characteristics are related to user performance and can be quantified mainly using objective measures. The last two sub-characteristics are related to the perception of the end-user or evaluator using the Web Application and can be quantified mainly using subjective measures.

The former three sub-characteristics were decomposed into other sub-characteristics or measurable attributes, taking into account the ergonomic criteria proposed in Bastien and Scapin [4]:

i. *Learnability* refers to the attributes of a Web application that facilitate learning: a) *help facilities* such as on-line

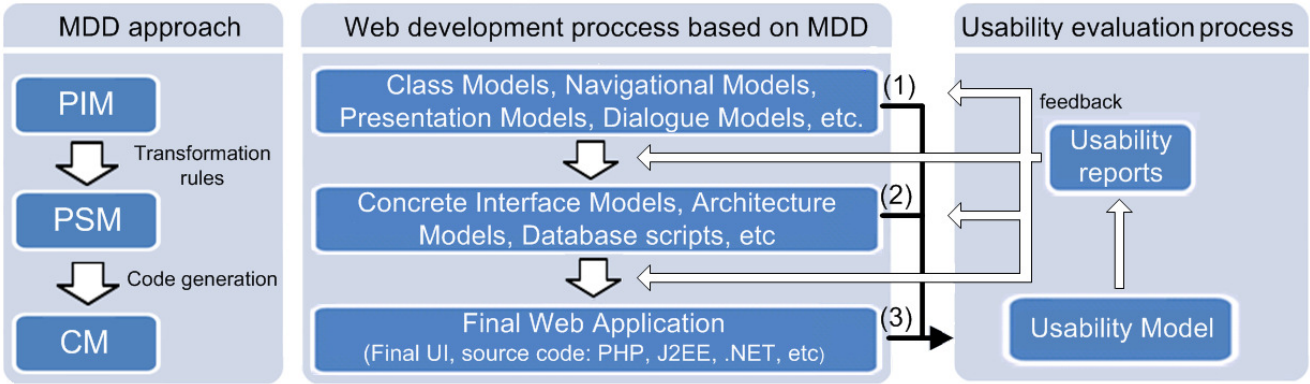

Fig. 1 Integrating a Usability Evaluation Process into a Model-Driven Web development process

help, contact section, etc; b) *predictability*, which refers to the ease with which a user can determine the result of his/her future actions (i.e. significance of link/image titles); c) *informative feedback* in response to user actions; and d) *memorability* as a measure of how quickly and accurately users can remember how to use a Web application that they have used before.

ii. *Understandability* refers to the attributes of a Web application that facilitate understanding: a) optical *legibility* of texts and images (e.g., font size, text contrast); b) *readability*, which involves aspects of information-grouping cohesiveness and density; c) *familiarity*, the ease with which a user recognizes the user interface components and views their interaction as natural; d) *brevity*, which is related to the reduction of user cognitive effort; and finally, e) *user guidance*, which is related to message quality, immediate feedback (to show the current user state), and navigability (to guide the user and to improve the access to the Web content).

iii. *Operability* refers to the attributes of a Web application that facilitate user control and operation: a) *execution facilities* such as compatible browsers, plug-ins needed, and update frequency; b) *data validity* of the user inputs; c) *controllability* of the services execution such as cancel, undo and redo support; d) *capability of adaptation* which refers to the capacity of the Web application to be adapted to the users' needs and preferences and e) *consistency* in the execution of services and control behavior.

The last two sub-characteristics are related to the perception of the end-user (*attractiveness*) or evaluator (*compliance*) using the Web Application:

iv. *Attractiveness* refers to the attributes of a Web application that are related to the aesthetic design. They can be quantified by measuring the UI uniformity in terms of *font style* (color, face and size), *background color*, and *position of elements*.

v. *Compliance* can be measured by assessing the agreement of the proposed Web Usability Model with respect to the standard SQuaRE and several Web design style guides.

Once the sub-characteristics have been identified, Web metrics are associated to the measurable attributes in order to quantify them. Values obtained from these Web metrics will allow us to interpret if measurable attributes contribute to achieving certain usability level in the Web application. The metrics included in our model were mainly extracted from the survey presented in Calero *et al.* [5]. We only selected those metrics that were theoretically and/or empirically validated. In addition, we proposed new metrics for several measurable attributes that were not appropriated covered by this survey.

As an example, we show some definitions of new proposed metrics that can be associated to attributes of the Web Usability Model:

- *Number of different font styles for textual links:* This metric is defined as number of different font style combinations (size, face, and color) for all textual links in the same navigation category. (Scale type: absolute value greater or equal to 1). The interpretation is: more than one style combination in the same navigation category means that font style uniformity is not insured. This metric is associated to the *font style uniformity* attribute, which belongs to the *attractiveness* sub-characteristic (iv).

- *Proportion of elements that show current user state*: This metric is defined as the ratio between the number of elements that show feedback about the current user state and the total number of elements that are required to have this feedback capability. (Scale type: ratio between 0 and 1). The interpretation is: values closer to 1 indicate that user can obtain feedback about his/her current state in the Web application. This metric is associated to the *immediate feedback* attribute, which belongs to the *user guidance* sub-characteristic (ii. e).

Web metrics definitions from the Web Usability Model are generic, and their calculation formula must be instantiated by identifying variables from this formula in the modeling primitives of the concrete Web development method for each abstraction level (PIM, PSM or CM). Not all the metrics can be defined at all the abstraction levels, for instance, the former metric can be applied at PIM level (if style properties are defined at the abstract UI) or at CM level (if style properties are defined in Cascading Style Sheets files). However, the second metric only can be defined at PSM or CM level since the feedback depends on the widget behavior from the concrete interface.

5. INSTANTIATION OF THE USABILITY EVALUATION PROCESS

In this section, we show an overview of how the previous usability process can be instantiated into a concrete Web development methodology. As an example, we selected the OO-H [7] method.

The OO-H method [7] provides designers with the semantics and notation for developing Web applications. The set of conceptual models that represents the different concerns of a Web application are: the specification of content requirements (Class Model) and the specification of functional requirements in terms of navigation needs (Navigation Model, NAD). A merge between the class and navigation models results in an Abstract Presentation Diagram as an integrated PIM model, which presents an abstract user interface as a collection of abstract pages. APD can be refined by a pattern catalog. Finally, platform-specific models (PSMs) are automatically obtained from the APD, from which source code (CM) can be automatically generated.

Next, we show as an example, a brief description about the steps involved in our usability evaluation process.

Step 1 (See Section 3): The purpose is to evaluate the usability of a Web application developed following the OO-H method. The selected Web application is a task management system developed for a Web development company located in Alicante, Spain. Finally, the attributes chosen were *font style uniformity* to evaluate the *attractiveness* sub-characteristic, and *immediate feedback* to evaluate the *user guidance* sub-characteristic, at least to some extent.

Step 2 (See Section 3): The artifacts selected for this evaluation were the final UIs (Code Model). The metrics selected to evaluate the previous attributes were *Number of different font styles for text links* and *Proportion of elements that show current user state* (see explanation of each metric in Section 4). The rating level for the former metric was established at *no UP* for values equal to 1; *low UP* for values equal to 2; *medium UP* for values equal to 3; and *critical UP* for values greater than 3. The rating level for the second metric was established at *no UP* for values equal to 1; *low UP* for values in the range [0.8, 1};

medium UP for values [0.5, 0.8} and *critical UP* for values [0, 0.5}. The usability report is defined as a list of usability problems (UP) detected with the next fields: description of the UP, level of the UP (critical, medium, or low), source of the problem (model), occurrences, and recommendations to correct it. More fields can be defined such as priority, impact, etc.

Step 3 (See Section 3): In this case, the evaluator is the same developer. The task assigned was the evaluation of all the user interfaces (CM) in order to present a usability report which will contain the usability problems detected with all the proposed fields filled in.

Step 4 (See Section 3): As an example, we only show the execution of the evaluation of one user interface (CM). Figure 2 shows a user interface automatically generated (Code Model) that represents the task management functionality of the Web application.

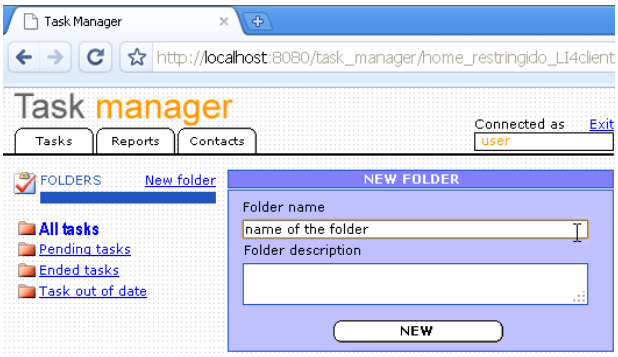

Fig.2 Example of a User interface automatically generated (Code Model)

The selected metrics were applied:

1. *Number of different font styles for textual links* [1]: The textual links that appears in the user interface (Fig. 2) are *Tasks*, *Reports*, *Contacts* and *Exit* from the top menu; and *New Folder*, *All tasks*, *Pending tasks*, *Ended tasks*, and *Tasks out of date* from the left menu. In the first navigation category (top menu), the value of the metric is 2 since the links *Tasks*, *Reports*, *Contacts* are displayed in a different style from the *Exit link*, which is displayed in a different color and it is also underlined. In the second navigation category (left menu), the value of the metric is also 2 since the links *New Folder*, *Pending tasks*, *Ended tasks*, and *Tasks out of date Contacts* are displayed in a different style from the *All tasks*, which is displayed in a different font face and font size. The rating level of the metric (see Step 2) indicates the existence of a low usability problem (UP001) for each menu.
2. *Proportion of elements that show current user state* [1]: The user interface must show the current user state, it means, the current section and the current task that is being performed. There are three types of elements that show the current user state in the Web application: the tabs from the top menu (*Tasks*, *Reports,* and *Contacts)*; the shape changes of the cursor when it is pointing on a textbox; and the highlight effects of a textbox when it has focus. Since the tabs are the only type of element that does not explicitly show the section in which the user is currently interacting, the value of the metric is 2/3=0.66. The rating level of the metric (see Step 2) indicates the existence of a medium usability problem (UP002).

Steps 5 and 6 (See Section 3): The usability problems detected after applying the previous metrics, can be explained in a usability report that contains the UP001 (See Table 1) and the UP002 (See Table 2).

Table 1. Usability problem detected: UP001

id	UP001
Description	The links *Tasks*, *Reports*, and *Contacts* are displayed in a font style that is different from the font style of the *Exit* link. The same problem occurs with the *all tasks* link that is displayed in a font style that is different to the used in the links: *New Folder*, *Pending tasks*, *Ended tasks*, and *Tasks out of date.*
Affected attribute	Attractiveness / font style uniformity.
Level	Low (rating level: 2).
Source of the problem	Abstract Presentation Diagram (PIM model).
Occurrences	2 occurrences (top menu and left menu)
Recommendations	Change the font style properties for the links *Tasks*, *Reports*, *Contacts* and *all tasks* in the Abstract Presentation Diagram. In this PIM model font style properties can be defined.

Table 2. Usability problem detected: UP002

id	UP002
Description	Tabs do not show the current user state in the Web application.
Affected attribute	Understandability/ User Guidance/ Immediate feedback.
Level	Medium (rating level: 0.66)
Source of the problem	The transformation rule that maps the representation of the tabs: *Task, Reports* and *Contacts* (PIM level) with the specific widget of the platform that shows the tabs (PSM).
Occurrences	1 occurrence for each UI that shows these tabs.
Recommendations	The widget target of the transformation rule should be changed for other widget with a highlight feature when a tab is clicked.

After analyzing and applying the proposed recommendations, a more usable Web application can be obtained without to need maintenance of source code.

6. DISCUSSIONS AND FUTHER WORK

This paper has presented a proposal in progress to integrate a usability evaluation process into Model-Driven Web development processes. The purpose of our work is to give an outline of a generic usability evaluation process supported by a Web Usability Model. A preliminary version of a usability evaluation process supported by a Web usability Model has been presented. Our Web Usability Model decomposes the

[1] It should be note that both metrics must be applied to all the user interfaces of the entire Web application.

usability sub-characteristics (from the Software Quality Model proposed in the ISO/IEC 25000 SQuaRE standard) into other sub-characteristics and measurable attributes taking into account ergonomic criteria. Web metrics were associated to measurable attributes in order to quantify them. Finally, a brief example has been shown in order to illustrate how the usability evaluation process can be instantiated into a concrete Web development method that follows the MDD approach. Although our example only shows a CM evaluation providing feedback to PIM models or transformations between PIM and PSM models, the usability evaluation process can evaluate intermediate artifacts (PIM and PSM models) by selecting metrics that their calculation formula has been defined to be applied to concrete PIM and PSM models (i.e., depth and breadth of a navigational map [1] associated to the navigability attribute).

We believe that the inherent features of model-driven development processes (e.g., traceability between models by means of model transformations) provide a suitable environment for performing usability evaluations. Specifically, if the usability of an automatically generated user interface can be assessed, the usability of any future user interface produced by this approach could be predicted. In other words, we are talking about a user interface that is usable by construction [2], at least to some extent.

In this way, usability can be taken into account throughout the entire Web development process, enabling Web applications to be developed with better quality thereby reducing effort at the maintenance stage.

Further work is intended to:

- Perform an entire instantiation of the usability evaluation process into the OO-H method.
- Define guidelines in order to guide evaluators on how the Web Usability Model can be applied
- Explore aggregation mechanisms for aggregating values obtained by individual metrics, and perform analyses of the impact on how the attributes affect (negatively or positively) other attributes of the Web Usability Model.
- Instantiate the evaluation process into different Model-Driven Web Development methods in order to improve our approach.
- Develop a tool to support the entire usability evaluation process. The tool will be able to manage the Web Usability Model by creating a repository of catalogued metrics following the SQuaRE patterns.

7. ACKNOWLEDGMENTS

This work is financed by META project (ref. TIN2006-15175-C05-05), the Quality-driven Model Transformation Project from the Universidad Politécnica de Valencia. The authors thank Jaime Gomez from Universidad de Alicante for his valuable help in providing the generated Web application and its models used to illustrate our usability evaluation process.

8. REFERENCES

[1] Abrahão, S., Condori-Fernández, N., Olsina, L., and Pastor, O. 2003. Defining and Validating Metrics for Navigational Models. Proc. of the 9th Inter. IEEE Software Metrics Symposium, 200-210.

[2] Abrahão, S., Iborra, E., and Vanderdonckt J. 2007. Usability Evaluation of User Interfaces Generated with a Model-Driven Architecture Tool. Maturing Usability. Springer HCI series, Vol. 10, 3-32.

[3] Abrahão, S. and Insfran, E. 2006. Early Usability Evaluation in Model-Driven Architecture Environments. Proc. of the 6th IEEE International Conference on Quality Software. IEEE Computer Society, 287-294.

[4] Bastien, J.M. and Scapin, D.L. 1993. Ergonomic criteria for the evaluation of human-computer interfaces. Tech. Rep. num.156. INRIA, Rocquencourt, France.

[5] Calero C., Ruiz J., and Piattini M. 2005. Classifying Web metrics using the Web quality model. Emerald Group Publishing Limited. Vol. 29(3), 227-248.

[6] Ceri, S., Fraternali, P., and Bongio, A. 2000. Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. Proc. of the 9th WWW Conference, 137-157.

[7] Gómez, J., Cachero, C., and Pastor, O. 2001. Conceptual Modeling of Device-Independent Web Applications. IEEE MultiMedia, Vol. 8(2), 26-39.

[8] ISO/IEC. 1998. ISO/IEC 9241-11, Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs), Part 11: Guidance on Usability.

[9] ISO/IEC. 2001. ISO/IEC 9126-1 Standard, Software Engineering, Product Quality - Part 1: Quality Model.

[10] ISO/IEC. 2005. ISO/IEC 25000 series, Software Engineering, Software Product Quality Requirements and Evaluation (SQuaRE).

[11] Kraus, A., Knapp, A., and Koch, N. 2006. Model-Driven Generation of Web Applications in UWE. 3rd Inter. Workshop on Model-Driven Web Engineering.

[12] Moraga, M.A, Calero, C., Piattini, M., and Diaz, O. 2007. Improving a portlet usability model. Software Quality Control, Vol. 15(2), 155-177.

[13] Olsina, L. and Rossi, G. 2002. Measuring Web Application Quality with WebQEM. IEEE Multimedia, Vol. 9(4), 20-29.

[14] Panach, J., Condori-Fernández, N., Valverde, F., Aquino, N., and Pastor, O. 2007. Towards an Early Usability Evaluation for Web Applications. International Conference on Software Process and Product Measurement. LNCS Springer Vol. 4895, 32-45.

[15] Seffah, A., Donyaee, M., Kline, R.B., and Padda, H.K. 2006. Usability Measurement and Metrics: A Consolidated Model. Software Quality Journal, Vol. 14(2), 159-178.

[16] Ivory, M.Y. 2001. An Empirical Foundation for Automated Web Interface Evaluation. PhD Thesis. University of California, Berkeley, Computer Science Division.

Playability as Extension of Quality in Use in Video Games

J. L. González Sánchez
Software Engineering Dept.
University of Granada
C/Periodista Daniel Saucedo Aranda s/n E-18071
(Granada–Spain)
+34 958 242 812
joseluisgs@ugr.es

F. Montero Simarro
Software Engineering Dept.
University of Castilla - La Mancha
Campus Universitario s/n E-02071 (Albacete–Spain)
+34 967 599 200 Ext.: 2468
fmontero@dsi.uclm.es

N. Padilla Zea
Software Engineering Dept.
University of Granada
C/Periodista Daniel Saucedo Aranda s/n E-18071
(Granada–Spain)
+34 958 240 849
npadilla@ugr.es

F. L. Gutiérrez Vela
Software Engineering Dept.
University of Granada
C/Periodista Daniel Saucedo Aranda s/n E-18071
(Granada–Spain)
+34 958 242 812
fgutierr@ugr.es

ABSTRACT
The quality of a software product is a main objective that every interactive system should aspire. There are many challenges to achieve this quality that require a previous characterization to ensure it. The International Standards Quality Models help to characterize the quality of a software system. But, there are some products that present 'special' quality requirements. In this paper we focus on special interactive systems: Video Games, whose quality requirements are different than traditional software. This additional dimension is called 'Playability'. In this paper, an extension of Quality in use Model for Playability decomposition (PQM) is introduced. In our playability quality model metrics are also considered and interpreted. Finally, we review different usability evaluation methods in order to identify what are the best evaluation methods for supporting playability evaluation tasks.

Categories and Subject Descriptors
H.1.2 [**Information Systems**]: User/Machine Systems - *Human factors*

General Terms
Design, Experimentation, Human Factors.

Keywords
Quality in Use, Interactive Systems, Video Games, Playability, Usability, User Experience.

1. INTRODUCTION
The Interactive Software Federation of Europe (ISFE) reveals new research findings that video games and entertainment systems collectively make up the biggest industry in terms of turnover, more so than music and cinema. We can deduce that videogames have become the preferred game of choice, exerting significant social and cultural influence over children, teens and adults [18]. As the quality of software has a direct bearing on product success and the User Experience, it should be taken into account throughout product development (hardware or software), so as to achieve the optimum experience for the player. The importance of video games in the actual society justifies the need to ask if the means of quality in this type of software is similar from the definition of the interactive or desktops software quality definition to guarantee an optimal User Experience.

In this work, we analyze how the game experience presents characteristics that are not explicitly in the quality standards models and why the usability or *quality in use* is not sufficient in video games context. We present a quality in use model for video games using playability to extend it for entertainment systems, with different attributes, facets and metrics to characterize the player experience with videogames.

2. THE QUALITY IN A SOFTWARE PRODUCT
When a *Desktop System* (DS) or Traditional Interactive System, such as a word processor, is developed, the main objective is that users can execute a set of tasks in a predetermined context, for example working in an office. The quality of this kind of systems has two main components: The first covers the functional aspects (functional utility) with two points of view: internally and externally. It has focused on disciplines such as Software Engineering. Another component indicates the means by which users can achieve this functionality. It is denominated *Usability* which has a great importance in HCI discipline. Usability represents a measure of product use whereby users achieve concrete objectives within a specific context of use.

Usability has been characterized in different international standards. ISO 9241-11:1998 [13] presents and define the Usability only as a characteristic of the process of use. In ISO/IEC 9226-1:2001[11] usability appears integrated in the properties of any software product. But, it is important to remark that the means of usability in the different standards models is not the same. In the first standard usability is: effectiveness, efficiency and satisfaction. But, in the second it is the easy of learning, understanding, operability and the attractiveness when use a software system.

These discrepancies between the standards are present in the following standards models. In ISO/IEC TR 9126-4: 2004 [12] appears the concept denominated *Quality in Use* whose definition is the same as the usability, but add the attribute of security.

Recently, ISO/IEC 25010:2009 [10] makes its contribution in this direction. The quality of a software system is described in terms of its elements and the interaction process. In this standard the Usability it is not one of the quality factor, it is an attribute of the Quality in Use with the flexibility and the security and they are associated to the interaction or process of use. Accepted recommendation in user interfaces design to improve the user experience can be found in [17, 22].

3. THE QUALITY IN VIDEO GAMES
The researches in HCI context have centred their objectives to study the user's abilities and cognitive process forgetting the

emotional dimension. A new concept, which is called *User Experience* (UX) [9], appears with this dimension. In entertainment systems it is only a partial vision of the reality, because it does not take into account all the quality attributes that influence the use of this 'special' interactive systems. These attributes identify the Player Experience (PX).

As we remarked previously, a videogame can be considered a 'special' interactive system, in that it is used for leisure purposes by users seeking fun and entertainment. Whereas the purpose of a desktop system is to execute a task, determined by a clear functional objective, our objectives when playing a videogame are more likely to be diverse and subjective. A videogame is not conceived for the user to deal with daily tasks, but rather it has a very specific objective: to make the player feel good when playing it. This objective is more subjective and personal than traditional software. Important recommendation for designing entertainment systems, based on this idea, can be found in [15, 21].

We propose that analyzing the quality of a videogame purely in terms of its Usability or Quality in Use is not sufficient – we need to consider not only functional values but also a set of specific non-functional values, given the properties of videogames. Additional factors to be considered might include, for example: rules of play; goals; storytelling techniques; virtual world recreation; character design, and so on. In other words, the PX could be much more complex than the UX. Hence we need to establish a set of attributes and properties to identify and measure the experience of players playing a videogame. These properties indicate to us whether a game is 'playable' or not – that is, they will identify the *Playability* of the video game. Later, we can use its properties to ensure the quality of a video game through a process led by playability goals to improve experience when players play the videogame, PX. In Table 1 we present the differences between some goal to achieve in the design of an optimal User Experience and Player Experience [16].

Playability is a live topic in the scientific community; it has been studied from different points of view and with different objectives without consensus on its definition or the elements that characterise it. We have identified two specific strands of research: *Playability* as *only Usability* in video games context (understanding and control of the game system), and research based on particular elements of video games [5, 15]. In the second line of research, we find references to: *Playability* in the quality of game elements [16, 20]. There are few studies focused on defining *Playability* formally, [4, 14], but without specific reference to *Playability* attributes or properties to characterize it. *Playability* is based on Usability, but in the context of video games, goes much further. Furthermore, *Playability* is not limited to the degree of 'fun' or 'entertainment' experienced when playing a game. Although these are primary objectives, they are concepts so *subjective*. It entails to extend and complete formally the User Experience characteristics with *players' dimensions* using a broad set of attributes and properties in order to measure the *Player Experience*.

In previous works, González Sánchez et al [6, 7, 8] proposed the characterization of the Player Experience with a video game based on Playability (PM, Playability Model), showing which attributes and examples of their properties are needed to analyze the 'game experience'. They present a conceptual framework for analysis of player experience and its relationship with the most common elements that may form part of video game architecture.

Table 1. Different objectives between UX and PX Design

	UX Usability Goals: Productivity		**PX Playability Goals: Entertainment**
1.	Task completion	1.	Entertainment
2.	Eliminate errors	2.	Fun to beat obstacles
3.	External reward	3.	Intrinsic reward
4.	Outcome-based rewards	4.	Process is its own reward
5.	Intuitive	5.	New things to learn
6.	Reduce workload	6.	Increase workload
7.	Assumes technology need to be humanized	7.	Assumes humans need to be challenged

4. PLAYABILITY AS QUALITY OF GAME EXPERIENCE

To characterize the quality of game experience we will make use of a precise and complete analysis of Playability, attributes, and a conceptual framework to evaluate it in any video game, either from the viewpoint of the game as an interactive process or from the player who performed/plays with it [7, 8]. This characterization must be coherent with existed standard, especially the most recent because we understand that they are the most consensual and complete.

As we have remarked, the *quality of a software product* has two main points to be analyzed: the *quality of process* and the *quality of product*. We need to consider additional aspects related to the user experience/player, which are related to the emotional aspects of interaction with video games.

In [8] we defined Playability as:

'a set of properties that describe the Player Experience using a specific game system whose main objective is to provide enjoyment and entertainment, by being credible and satisfying, when the player plays alone or in company'.

It is important to emphasise the 'satisfying' and 'credible' dimensions. The former is more difficult to measure in video games than in desktop systems due to the high degree of subjectivity of non-functional objectives. Similarly, the latter depends on the degree to which players assimilate and become absorbed in the game during play – also difficult to measure objectively with traditional usability test. The Definition of Playability can be based on Quality in Use, but it should be added the above attributes. Also, the definition of particular properties or Quality in Use must be rewriting. For example 'Effectiveness' in a video game is not related to the speed with which a task can be completed, because typically a player will play for entertainment and relax, this being one of the game's main objective. With all of these considerations, Playability represents

'the degree in which specific player achieve specific game goals with effectiveness, efficiency, flexibility, security and, especially, satisfaction in a playable context of use.'

In Fig. 1 we present our *Playability Quality Model* (PQM) as an extension of the Quality in Use model ([2, 10]). It is focus on video games software applications. Next each quality factor and attribute in our quality model will be defined following the previously mentioned ISO standard.

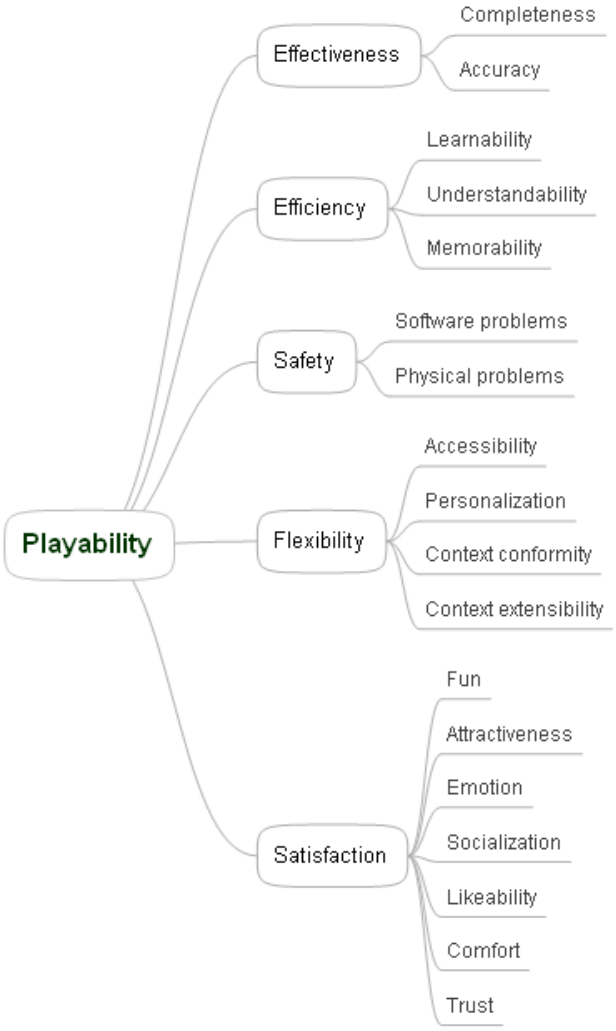

Fig. 1. Quality model for playability

- ***Effectiveness:*** We define the degree to which specific users (players) can achieve the proposed goals with precision and completeness in the context of use, the video game.
- ***Efficiency:*** It is the degree to which specific users (players) can achieve the goals proposed by investing an appropriate amount of resources in relation to the effectiveness achieved in a context of use, the video game. This factor is determined by the ease of learning and immersion.
- ***Flexibility:*** It is the degree to which the video game can be used in different contexts or by different player or game profiles.
- ***Safety:*** It is acceptable level of risk to the player health or data in a context of use, the video game.
- ***Satisfaction:*** It is the degree to which users (players) are satisfied in a context of use, the video game. In this factor we consider various attributes such as fun, attractiveness, motivation, emotion or sociable.

Playability analysis is a very complex process due to the different perspectives that we can use to analyze the various parts of video game architecture. In this work, we propose a classification of these perspectives based on six *Facets of Playability* (PF). Each facet allows us to identify the different attributes and properties of *Playability* that are affected by the different elements of video game architecture [7]. The six *Facets of Playability* are:

- ***Intrinsic Playability*:** This is the *Playability* inherent in the nature of the videogame itself and how it is presented to the player. It is closely related to Game Core.
- ***Mechanical Playability*:** This is related to the quality of the videogame as a software system. It is associated to the Game Engine
- ***Interactive Playability*:** This is associated with player interaction and videogame user interface development. It is strongly connected to the Game Interface.
- ***Artistic Playability*:** This facet relates to the quality of the artistic and aesthetic rendering in the game elements (visual graphics, melodies, storyline and storytelling).
- ***Intrapersonal Playability*** or *Personal Playability*: This refers to the individual outlook, perceptions and feelings that the videogame produces in each placer and as such has a high subjective value.
- ***Interpersonal Playability*** or *Social Playability*: This refers to the feelings and perceptions of users, and the group awareness that arise when a game is played in company, be it in a competitive, cooperative or collaborative way.

The overall *Playability* of a videogame, then, is the sum total of values across all attributes in the different *Facets of Playability*. It is crucial to optimize *Playability* across the different facets in order to guarantee the best Player Experience.

5. PLAYABILITY AS MEASURE OF QUALITY IN A VIDEO GAME

We complete Quality in Use model based on Playability with the identification and association of metrics to the identified factors and attributes. To approach this task we use the international standards and we have adapted the different metrics and measures to evaluate and test video games.

The metrics, Table 2, have as objective the estimation of the quality of Player Experience with video games. Each column reflects the characterization of the different identified metrics. These characteristics are: the name of the metric, the objective that we analyze with it, its formula, the interpretation of the numerical value and the type of evaluation to estimate its value. We must to remark all the indentified metrics are focused in the use of the video game. Hence, the evaluation essentially requires test with players, observation to players when are playing and in players' satisfaction case the realization of questionnaires when they complete the playtime.

Playability evaluation is related to evaluation of the user's performance and satisfaction when using the game, product or system in a real or simulated entertainment environment.

In this paper, see Table 2, we identified many relationships between playability and quality in use metrics, and we think that quality in use metrics are useful for playability evaluation. But some metrics should be interpreted in a different manner. For instance, if we have traditional software products, effectiveness metrics in international standards introduce *tasks effectiveness* or *task completion* as metrics. But when a game and playability is considered, we need to speak in terms of 'goals' in entertainment game context, as the challenges that the game introduced.

In a similar manner, *error frequency* metric in traditional software has sense, and a value closer to 0 is the better, but in games we propose *attempt frequency* as metric, and we can find values closer to 0 if expert players are playing, and closer to 1 if novice or clumsy players are considered. Normally, games introduce difficulties to capture and suck new players; a very simple game is not attractive, because it will be bored.

The *personalization* is an advisable factor in video games because in this software exists many design elements that try to distract, and to accompany the form of interaction. It should be flexible, for example supporting different interaction techniques: keys, pads, controls, menus, sounds and so on. The attribute of *accessibility*, however desirable and enforceable, traditionally has not enjoyed much attention in the development of video games. Nowadays this is changing and the presence of this attribute contributes to the use of it in the video game interface and mechanics.

Table 2. Metrics associated to playability attributes

	Metric name	Purpose	Formula	Interpretation	Evaluation method
Effectiveness	Goal effectiveness	What proportion of the goals is achieved correctly?	$M1 = \|1-\Sigma A_i\|$ A_i proportional value of each missing	M1 ∈ [0, 1], the closer to 1 the better	User test
	Goal completion	What proportion of the goals are completed?	X = A/B A = n. of goals completed B = total number of attempted goals	M1 ∈ [0, 1], the closer to 1 the better	User test
	Number of attempt	What is the frequency of attempts?	X = A A = n. of attempts made by the player	Expert player closer to 0. At the beginning > 0	User test
Efficiency	Goal time	How long does it take to complete a goal?	$X = T_a$	Novice players will have more time	User test
	Goal efficiency	How efficient are the users?	X = M1/T	X ∈ [0, 1], closer to middle value	User test
	Relative user efficiency	How efficient is a player compared to an expert?	X = A/B A = ordinary player's goal efficiency B = expert player's goal efficiency	M1 ∈ [0, 1], the closer to 1 the better	User test
Flexibility	Accessibility	What proportion of the goals can be achieved by using alternative ways of interaction?	X = A/B A = goals with alternative interactions B = total number of goals	M1 ∈ [0, 1], the closer to 1 the better	User test
	Personalization	What proportion of the personalization options are used by the players?	X = A/B A = personalized elements B = elements in the game	M1 ∈ [0, 1], if closer to 1 original interaction way, perhaps should be changed	User test
Safety	User health and safety	What is the incidence of health problems among users of the product?	X = 1 – A / B A = number of players reporting problems B = total number of players	M1 ∈ [0, 1], the closer to 1 the better	User test
	Software damage	What is the incidence of software corruption?	X = 1 – A / B A = n. occurrences of soft. corruption B = total number of usage situations	M1 ∈ [0, 1], the closer to 1 the better	User test
Satisfaction	Satisfaction scale	How satisfied is the player?	X = A/B A = questionnaire producing psychometric scales B = population average	X>0 the larger the better	User test + questionnaires
	Satisfaction questionnaire	How satisfied is the user with specific software features?	$X = \Sigma A_i /n$ A_i= response to a question B = number of responses	Compare with previous values, or with population average	User test + questionnaires
	Discretionary usage	What proportion of potential users choose to use the system?	X = A/B A = number of times that specific software functions are used B = number of times players are intended to be used	M1 ∈ [0, 1], the closer to 1 the better	Observation of usage
	Socialization	What proportion of potential users choose to use the system?	X = A/B A = number of times that game is used in a collaborative environment B = number of times that game is used	M1 ∈ [0, 1], the closer to 1 collaborative game, closer to 0 personal game	Observation of usage

Accessibility is a quality attribute considered in the definition of quality in use. In our playability model proposal, that attribute is also considered. Accessibility problems can be considered to be usability problems for particular group of players e.g. those with disabilities. If a player cannot understand what is said in cut scenes or cannot hear the footsteps of someone sneaking up behind him or her, because the player suffers from an auditory disability or if the game does not support the use of specific input devices such as one handed controllers or sip and puff joysticks that allow severely physical disabled players to play the game.

The *safety* is an important factor nowadays in video games. The game process is not only a static and mental activity. In some cases, it demands physical requirements, for example game controls that demands and important corporal or physical effort and their effects can be sometimes potentially dangerous or not very recommendable to the player health if the player carries out this activity for a long time.

Satisfaction is the most important attribute in videogames due to different aspects can be considered in it: cognitive, emotional, physical, fun and social. The estimation of the degree of satisfaction in a video game is realized using questionnaires and observing the player during the game process and analyzing the user preferences in the different game sessions with video games. Probably, when games are considered the more important or determinant quality attribute is the achieved satisfaction rating. This attribute is subjective and in our playability quality model is enriched by using additional quality attributes and sub-attributes.

Thanks to proposed metrics, the quality model of the player experience with videogames based on playability, (PQM) is complete as [1] recommend for quality models developing.

In last column of Table 2 different *playability* evaluation methods are suggested for each metric. These evaluation methods are the same that we use for usability evaluation. In the next section, we will discuss different evaluation methods; our main goal will be use these methods for playability evaluation purposes.

6. PLAYABILITY EVALUATION METHODS

This section reviews usability evaluation methods (UEMs) gathered in different reports from MAUSE project. MAUSE project was a COST Action, COST 294 from 2004 to 2009. The ultimate goal of MAUSE was to bring more science to bear on UEM development, evaluation, and comparison, aiming for results that can be transferred to industry and educators, thus leading to increased competitiveness of European industry and benefit to the public. In this paper, we are focused on another quality factor; playability and we want to discuss if UEM are useful as playability evaluation.

In COST 294, four major research and development activities were implemented by four working groups. Concretely, working group 1 did a critical review and analysis of individual UEMs. The primary goal of this activity was to build a refined, substantiated and consolidated knowledge-pool about usability evaluation, based on the expertise, experiences, and research works of the participating project partners. Different reports were written and [19] were used in this paper as input.

In order to evaluate previous proposed metrics and quality model we need to specific playability evaluation methods (PEMs). In [19] three categories of evaluation methods were gathered: Data gathering and modeling methods (DGMM), User Interactions evaluation methods (UIEM), Collaborative methods (CM) and Mixed methodologies (MM),

First group, DGMM, is used for gaining knowledge about users and their activities. Two subcategories were distinguished: Data gathering methods (DGM) and Modeling methods (MM). These evaluation methods are useful for playability evaluation, but not always. Surveys and questionnaires come from social sciences, where surveys are commonly used and questionnaires are methods for recording and collecting information. In this context, games can be used by many kinds of user, for instance preschool children; 2 to 5 years old, surveys and questionnaires useful because it is also for them to verbalize their options. Think-aloud protocol is not a solution, because even school children ages 6 to 10 years may have difficulty with concurrent thinking aloud and they cannot be left alone.

Modeling methods (MM) are often associated with specific data gathering methods or their combination. In this set of methods, an example is especially interesting, Personas [3]. It is a precise descriptive model of the user, what user whishes to achieve and why. But this method is more a User-Centered Design complement. We think that other techniques associated, such as ConcurTaskTrees (CTT) or K-Made, are not useful when playability is considered. Normally, games need very complex models, because they have many interaction freedom degrees; games and activities for entertainment are rich interactive applications, where users can do things in many different ways.

Table 3. Heuristics and principles for game designing

(Korhonen and Koivisto, 2006)	(Rouse, 2001)
1. Don't waste the player's time.	1. Consistent World.
2. Prepare for interruptions.	2. Understand the Game-World's Bounds.
3. Take other persons into account.	3. Reasonable Solutions to Work.
4. Follow standard conventions.	4. Direction.
5. Provide gameplay help.	5. Accomplish a Task Incrementally
6. Differentiation between device UI and the game UI should be evident.	6. Be Immersed.
7. Use terms that are familiar to the player.	7. Fail.
8. Status of the characters and the game should be clearly visible.	8. A Fair Chance.
9. The Player should have clear goals.	9. Not Need to repeat themselves.
10. Support a wide range of players and playing styles.	10. Not Get Hopelessly Stuck.
11. Don't encourage repetitive and boring tasks.	11. To Do, Not to Watch.
	12. Do Not Know What They Want, But They Know It When They See It.

User Interaction Evaluation Methods (UIEM) are explicitly targeted towards evaluation. Knowledge-based and empirical methods are considered in this group. In these methods experts and experience is considered, but games are different from others kind of applications and heuristics or principles for them are not the same than Shneiderman [22] or Nielsen's principles [7]. In Table 3 some meaningful heuristics for game designing are shown [9, 10].

We think that user testing, observation and user testing (see Table 2 – 'Evaluation method' column) are the best manner in order to

playability evaluation. Many times these user testing are done with children and we must to know that tests cannot be done with children younger than 18 without the permission and supervision of their parents. Questionnaires are useful tool for playability evaluation too, but sometimes cannot be used, because children are too much young.

7. CONCLUSIONS AND FUTURE WORK

The quality of a system is the result of the quality of the system elements and their interaction. But every software applications are not equal. In this paper, games and entertainment software are considered. In this context, playability is our main quality measure and we presented a playability quality model based on international standard and the interaction component of the quality is especially taken into account.

We identified a direct connection between quality in use and playability. Quality in use is a useful concept when interaction with traditional software is evaluated. But games are different in many aspects from others kinds of software. In this paper, meaningful differences between games and traditional software in the quality model, metrics, and principles or heuristics were identified. In our proposal, the main contributions in playability characterization are related with the player's satisfaction and ISO/IEC 25010 [10, 19] was enriched in order to evaluate the interaction with games. Our metrics are ISO 9126-4 [12] inspired, but in this paper different interpretation and additional metrics are presented.

Nevertheless, these metrics need to be used and validated by using real games and evaluations experiments, and, in this moment, we are doing several evaluations in order to validate the proposed metrics.

8. ACKNOWLEDGMENTS

This research is financed by: the Spanish International Commission for Science and Technology (CICYT); the DESACO Project (TIN2008-06596-C02); and the F.P.U. Programme of the Ministry of Science and Innovation, Spain. Thanks to MAUSE project and COST nº. 294.

9. REFERENCES

[1] Basili, Victor R. and Weiss, D., "A Methodology for Collecting Valid Software Engineering Data", IEEE Transactions On Software Engineering, pp 728-738. Nov. (1984).

[2] Bevan, N. Extending Quality in use to provide a framework for usability measurement. Proceedings of HCI International 2009, San Diego, California, USA. (2009)

[3] Cooper, A. The Inmates are Running the Asylum. SAMS, (1999).

[4] Fabricatore, C., Nussbaum, M., Rosas, R.: Playability in Action Video Games: A Qualitative Design Model. Human-Computer Interaction, Vol 17, pp. 311-368 (2002).

[5] Federoff, M.: Heuristic and Usability Guidelines for the Creation and Evaluation of Fun Video Games. Master Thesis, University (2002).

[6] González Sánchez, J. L.; Gutiérrez, F. L.; Cabrera, M.; Padilla Zea, N.: Design of adaptative videogame interfaces: a practical case of use in special education. Computer-Aided Design of User Interfaces VI. Lopez Jaquero, V.; Montero Simarro, F.; Molina Masso, J.P.; Vanderdonckt, J. (Eds.). Springer (2009),

[7] González Sánchez, J. L.; Padilla Zea, N.; Gutiérrez, F. L.: From Usability to Playability: Itroduction to the Player-Centred Video Game Development Procces Proceedings of HCI International 2009, San Diego, California, USA. (2009)

[8] González Sánchez, J. L.; Padilla Zea, N.; Gutiérrez, F. L.; Cabrera, C.: De la Usabilidad a la Jugabilidad: Diseño de Videojuegos Centrado en el Jugador, In Proceedings of INTERACCION-2008, pp. 99-109. (2008).

[9] Hassenzahl, M.; Tractinsky, N.: User Experience – A Research Agenda. Behaviour and Information Technology, Vol. 25, N. 2. pp. 91-97 (2006).

[10] ISO/IEC 25010-3: Systems and software engineering: Software product quality and system quality in use models. (2009).

[11] ISO/IEC 9126-1: Software engineering – Product quality - Part 1: Quality model. (2001).

[12] ISO/IEC TR 9126-4: Software engineering – Product quality Part 4: Quality in use metrics. (2004).

[13] ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) Part 11: Guidance on Usability. (1998).

[14] Järvien, A., Heliö, S., Mäyrä, F.: Communication and Community in Digital Entertainment Services. Prestudy Research Report. Hypermeda Lab. University of Tampere (2002).

[15] Korhonen, H., Koivisto, E.: Playability Heuristic for Mobile Games. MobileHCI'06. ACM 1-59593-390-5/06/0009 (2006).

[16] Lazzaro, N.: The Four Fun Keys. Book Chapter in Game Usability. Advancing from the Experts for Advancing the Player Experience. Morgan Kaufmann, Burlington. (2008).

[17] Nielsen, J., Usability Engineering. Morgan Kaufmann, San Francisco. (1994).

[18] Provenzo, E.: Video kids. Cambridge: Harvard University Press (1991).

[19] R3UEMs: Review, Report and Refine Workshop (2007) Abstract: The current proceedings reflect the work conducted so far, and the contributions within WG1 of COST294-MAUSE project, as it will be discussed in its 3rd. International Workshop, Athens, March 5, 2007: "Review, Report and Refine Usability Evaluation Methods (R3 UEMs)". Dominique Scapin and Effie Law (Eds.)

[20] Rollings, A., Morris, D.: Game Architecture and Design. Ed. New Riders Games (2003).

[21] Rouse III, R. Game Design: Theory and practice. Wordware game developer's library. (2001).

[22] Shneiderman, B. Designing the user interface. Addison-Wesley Publishing. (1998).

Designing, Developing, Evaluating the Invisible? —
Usability Evaluation and Software Development in Ubiquitous Computing

Tom Gross
Faculty of Media
Bauhaus-University Weimar
Bauhausstr. 11, 99423 Weimar, Germany
+49 3643 58-3710
tom.gross(at)medien.uni-weimar.de

ABSTRACT
This position paper for the 2nd International Workshop on the Interplay between Usability Evaluation and Software Development (I-USED 2009) introduces some strengths of Ubiquitous Computing as well as some challenges it entails for the software development and usability evaluation; in particular it presents a user-centred design process for ubiquitous computing.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – Evaluation/Methodology; Prototyping; User-Centred Design.

General Terms
Human Factors.

Keywords
Software Development; Usability Evaluation; Ubiquitous Computing.

1. INTRODUCTION
Ubiquitous Computing (UbiComp) provides new opportunities and poses new challenges to software development and the usability evaluation. According to Mark Weiser who coined this term, UbiComp 'enhances computer use by making many computers available throughout the physical environment, while making them effectively invisible to the user' [11]. Instead of explicit input from devices such as a keyboard or a mouse, UbiComp systems typically get implicit input from users' interaction with their physical environment through everyday objects. Besides the advantages of the resulting invisibility and unobtrusiveness for the users, UbiComp entails a variety of challenges for their software development and usability evaluation.

2. CHALLENGES
The challenges that are mentioned in the literature include both the general unobtrusiveness [2], but also the complex interactions that make use of natural input technologies [2] with a great number of interaction partners [4] and through distributed devices [3] in a large physical space [4]. The fact that UbiComp is often seen as everyday computing, which is 'characterised by continuously present, integrative, and unobtrusive interaction' [1] induces further challenges such as highly mobile users [3, 5], interaction on small devices [3], timing difficulties through concurring interactions [10], and environmental factors that cannot be controlled [6].

3. SOLUTIONS
Methods from Human-Computer Interaction (HCI) have already been integrated into software development life cycles, but the process of finding and integrating designated methods into the UbiComp development life cycle is still in its early stages. In HCI, for instance, Jokela [8, 9] has extended the ISO 13407 standard 'ISO 13407: 1999 - Human-Centred Design Processes for Interactive Systems' [7]. This ISO 13407 regulates the design processes of the four phases: understanding and specifying the context of use, specifying the requirements, producing design results, and evaluating the design against the requirements in a loop from the first phase to the last, and then restarting with the first phase in an iterative cycle. We have extended and adapted this life cycle to fit to the specific needs of UbiComp (cf. Figure 1).

A general challenge in integrating methods into the design and development life cycle for UbiComp is to find or define natural and unobtrusive methods that reflect the nature and characteristics of UbiComp and everyday computing. In this 2nd International Workshop on the Interplay between Usability Evaluation and Software Development (I-USED 2009) workshop I would be particularly interested in discussing new approaches for the integration of usability concepts and methods into the software development processes—including traditional single-user systems, cooperative systems as well as particularly UbiComp systems.

4. CONCLUSIONS
Tom Gross is professor for Computer-Supported Cooperative Work and head of the Cooperative Media Lab at the Faculty of Media of the Bauhaus-University Weimar, Germany. His research interests include Computer-Supported Cooperative Work, Human-Computer Interaction, and Ubiquitous Computing. Since beginning of 2008 he is Prorektor (vice-president) of the Bauhaus-University Weimar. From 1999 to 2003 he was a senior researcher at the Fraunhofer Institute for Applied Information Technology FIT in St. Augustin, Germany. He holds a diploma and a doctorate degree in Applied Computer Science from the Johannes Kepler University Linz, Austria.

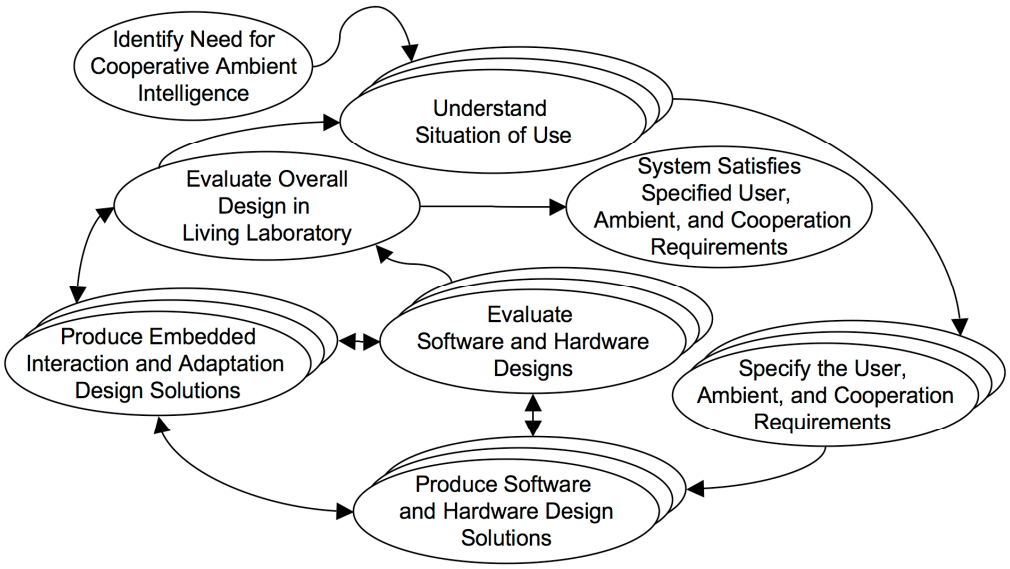

Figure 1. User-Centred Design Process for Ubiquitous Computing.

ACKNOWLEDGEMENTS

Thanks to the members of the Cooperative Media Lab—especially to Christoph Beckmann and Maximilian Schirmer.

REFERENCES

[1] Abowd, G.D. and Mynatt, E. Charting Past, Present, and Future Research in Ubiquitous Computing. ACM Transactions on Computer-Human Interaction 7, 1 (Sept. 2000). pp. 29-58.

[2] Carter, S. and Mankoff, J. Prototypes in the Wild: Lessons from Three Ubicomp Systems. IEEE Pervasive Computing 4, 4 (2005). pp. 51-57.

[3] Crabtree, A., Benford, S., Greenhalgh, C., Tennent, P., Chalmers, M. and Brown, B. Supporting Ethnographic Studies of Ubiquitous Computing in the Wild. In Proceedings of the 6th Conference on Designing Interactive Systems – DIS 2006 (Jun. 26-28, University Park, PA, USA). ACM, New York, NY, USA, 2006. pp. 60-69.

[4] Dey, A.K. Evaluation of Ubiquitous Computing Systems: Evaluating the Predictability of Systems. In Proceedings of the 3rd International Conference on Ubiquitous Computing – UbiComp 2001 (Sept. 30-Oct. 2, Atlanta, GA, USA). Springer-Verlag, 2001.

[5] Froehlich, J., Chen, M.Y., Consolvo, S., Harrison, B. and Landay, J.A. MyExperience: A System for In Situ Tracing and Capturing of User Feedback on Mobile Phones. In Proceedings of the 5th International Conference on Mobile Systems, Applications and Services – MobiSys 2007 (Jun. 11-14, San Juan, Puerto Rico). ACM, New York, NY, USA, 2007. pp. 57-70.

[6] Iachello, G., Truong, K.N., Abowd, G.D., Hayes, G.R. and Stevens, M. Prototyping and Sampling Experience to Evaluate Ubiquitous Computing Privacy in the Real World. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems – CHI 2006 (Apr. 24-27, Montreal, Quebec, Canada). ACM, New York, NY, USA, 2006. pp. 1009-1018.

[7] ISO. ISO 13407: 1999 - Human-Centred Design Processes for Interactive Systems. ISO - International Organisation for Standardisation, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=21197, 2008. (Accessed 24/4/2009).

[8] Jokela, T. Making User-Centred Design Common Sense: Striving for an Unambiguous and Communicative UCD Process Model. In 2nd Nordic Conference on Human-Computer Interaction - NordiCHI 2002 (Oct. 19-23, Aarhus, Denmark). ACM, N.Y., 2002. pp. 19-26.

[9] Jokela, T. A Method-Independent Process Model of User-Centred Design. In Hammond, J., Gross, T. and Wesson, J., eds. Usability: Gaining a Competitive Edge. Kluwer Academic Publishers, Dortrecht, NL, 2002. pp. 23-38.

[10] Kellar, M., Reilly, D., Hawkey, K., Rodgers, M., MacKay, B., Dearman, D., Ha, V., MacInnes, W.J., Nunes, M., Parker, K., Whalen, T. and Inkpen, K.M. It's a Jungle Out There: Practical Considerations for Evaluation in the City. In Extended Abstracts on Human Factors in Computing Systems – CHI 2005 (Apr. 2-7, Portland, OR, USA). ACM, New York, NY, USA, 2005. pp. 1533-1536.

[11] Weiser, M. Some Computer Science Issues in Ubiquitous Computing. Communications of the ACM 36, 7 (July 1993). pp. 75-84.

Bringing Usability Evaluation into Practice: Field Studies in Two Software Organizations

Jakob Otkjær Bak
TARGIT A/S
Aalborgvej 94
DK-9800 Hjørring
Denmark
jb@targit.com

Kim Nguyen
Logica
Fredrik Bajers Vej 1
DK-9220 Aalborg East
Denmark
kimmeren@gmail.com

Peter Risgaard
EUCNORD
Hånbækvej 50
DK-9900 Frederikshavn
Denmark
pri@eucnord.dk

Jan Stage
Aalborg University
Department of Computer Science
DK-9220 Aalborg East
Denmark
jans@cs.aau.dk

ABSTRACT
This paper explores how obstacles to usability evaluations in a software organization can be affected. We present two field studies, each conducted in a software organization that had no previous experience with usability evaluation. In each study, we first interviewed key stakeholders to identify their opinion about significant obstacles to conducting usability evaluations. Then we demonstrated the benefits of a usability evaluation by evaluating the usability of one of their software products, while being observed by the developers, and presenting the evaluation results to the developers. Finally, the key stakeholders were interviewed again to establish the effect of the demonstration. The demonstration of benefits had a positive effect on some of the key obstacles, while others were unaffected. One organization expressed future plans for conducting usability evaluations while the other was still reluctant.

Categories and Subject Descriptors
H.5.2. [**Information Interfaces and Presentation**]: User Interfaces – *Evaluation/methodology*. K.6.1 [**Management of Computing and Information Systems**]: Project and People Management – *Staffing, Systems development, Training*.

General Terms
Experimentation, Human Factors.

Keywords
Usability evaluation, software organizations, development practice, empirical study.

1. INTRODUCTION
Usability is a fundamental attribute of interactive systems [7], and it is critical to their success or failure on the market [10]. Evaluation of usability has been documented to be economically feasible because of increased sales [11], increased user productivity [12], decreased training costs [4] and decreased needs for user support [20]. Despite these facts, many software organizations are still not conducting any form of usability evaluation in their development process [21].

There have been considerable efforts to affect the obstacles that prevent these software organizations from deploying usability evaluation techniques. A major approach has provided techniques that are supposed to ease the deployment. This approach has only had limited success and mostly in software organizations that are already conducting usability evaluations. The reason may be that most of the proposed techniques are highly technical and designed by experts to be used by experts or at least by well-trained professionals [3].

A basically different approach has been to affect key stakeholders' attitudes to usability evaluation. This has mostly been done on a general level by documenting how other organizations have benefitted from deploying usability evaluation techniques in their development process. A study found that collection of user data, setting usability goals and conducting usability walkthroughs had a positive effect [13]. Another study documented that deployment of user-centered design in the development life cycle of a software company, specifically by integration of use cases in the development process, supported decision making [17].Karat provides evidence about the cost and benefit of usability evaluation [11]. The difficulty is, however, that often the cost is paid by the software organization, while the benefit is gained by the customer. Yet there are exceptions. A study established that evaluation of software for usability can lead to increased sale of products [12]. Another study demonstrated that the need for user support decreased with better usability [20]. Experience with deployment of usability work is usually focused on larger organizations. However, a study in a smaller organization also presents activities that were successful [5]. Another study focused specifically on usability evaluation and concludes that quick, cheap and effective evaluations can be conducted [19].

Only few studies have focused on affecting the attitudes to usability evaluation on a specific level; that is in a particular software organization. This paper reports from two field studies, where we tried to overcome obstacles to usability evaluation by affecting the attitudes of key stakeholders. This was done by demonstrating how that particular organization could benefit from deploying usability evaluation in their development process. In section 2 we present related work on affecting obstacles to usability evaluation. Section 3 presents the method used in of the two field studies. In section 4, we provide the results from the field studies. In section 5, we discuss our results. Finally, section 6 provides the conclusion.

2. RELATED WORK

The majority of studies that try to affect obstacles to usability evaluation focuses on usability guidelines and methods for incorporating usability in the development process. Gould and Lewis were among the first to provide guidelines for the deployment of usability in the design process [6]. A study questioned the relevance of guidelines to usability and discussed appropriate sources of guidance [3]. Overall guidelines directed at the developers are widely used. A study identified the gap between designers and users as the major obstacle to deploying usability and suggested usability engineering methodologies to help overcome this obstacle [23]. Grudin presented suggestions to overcome this gap based on long term experiences [8].

Solutions to overcome organizational obstacles to usability evaluation are presented in some papers. They tend to advise what usability practitioners can do to sell usability to the organization. Mayhew suggests three phases and for each phase how, why and what to do to sell usability [16]. A study concluded that communicating the message of usability is not enough; the facts must be solid and documented [24]. Another study complements this by concluding that experiences with usability have to be presented in a way that appeals to upper management's mindset with emphasis on the monetary benefits [1].

Resource-related obstacles have also been studied. Based on experiences from several organizations, Nielsen states that there are considerable monetary benefits from conducting usability evaluations [18]. A study emphasized that automation is a way to complement existing usability evaluation methods [9].

Only a few researchers have tried to measure the effect of deploying new usability methods in software organizations. One study concluded that nurturing the developers' skills in user-centered design was a major factor in developing more usable systems [22]. A different study provided a usability engineer to a software organization. This helped developers shift focus toward design and assume a role as the users' advocate [2].

3. METHOD

We have conducted field studies in two software organizations, where we tried to demonstrate the benefits of usability evaluation in an ongoing development process.

3.1 Company A

The company had, at the time of the study, 150 employees with headquarter in Denmark and branches in Canada, USA and Romania. Its business was separated in four units: supply chain solutions, postal solutions, airport solutions and care management solutions. Our collaboration was with the care management solution unit that had 12 employees, of which 7 were software developers. The system we evaluated was a planning module for a healthcare management system used by nurses and home assistants to plan both care for citizens and staff working hours. The system had been developed some years before and updated regularly. Initially, it had a non-graphical user interface. Later, it was supplemented with a graphical user interface.

The company's motivation for participating was curiosity about usability evaluation and a desire to see if it could be integrated in the development process without being too costly. It was not triggered by customer demands.

Participants. Three participants from company A were involved in the collaboration; a section manager, a developer and a user consultant. The section manager was in charge of the development team, the developer was responsible for the user interface design and the user consultant was responsible for contact to users and for their education.

Procedure. The study was conducted in 3 steps. The first step was an initial meeting with the section manager of the care management solutions department, the user consultant and the developer responsible of the user interface design. The purpose was to determine obstacles to usability evaluation in the company and select the part of the system to evaluate. After the meeting, the three participants were asked to write down weaknesses and obstacles to integration of usability in their development process.

The second step was the evaluation of the system. We used the Instant Data Analysis (IDA) method [14]. After the evaluation, the test results were emailed to the three participants and subsequently presented in combination with redesign proposals.

After a month, the third step was conducted. A meeting was held, where the developer and user consultant were interviewed about their experiences with the usability evaluation and its result. They were also asked if any changes had been made to the system or their work process. A telephone interview was conducted with the section manager who was asked the same questions.

Setting. The meetings were held in a conference room in the company. The usability tests were conducted with real users and took place at the users' workplaces. The user consultant and developer observed the first test session.

Data collection and analysis. We recorded of the interviews and collected the forms with opinions about weaknesses and obstacles. Each interview was conducted according to an interview guide [15]. Later, the recorded interviews were condensed using a method called "condensation of meaning" [15], and this outcome was then analyzed. The analysis was conducted by two persons separately. These two persons individually pointed out statements from the condensed data and grouped them into obstacles. Finally, the they negotiated a joint list of weaknesses and obstacles.

3.2 Company B

The company produced wireless technology. At the time of the study, it was divided into four units: technology, consumer products, network systems and healthcare. There were 180 employees, most of them located in the headquarter in Denmark. There were branches in USA, Hong Kong and Romania. Our collaboration with this company was carried out with the healthcare unit that had 10 employees, where 5 of them were developers. The system evaluated, was a device for home use by elderly people to send health data to a monitoring center. This system was recently developed and had a simple user interface.

The company's motivation for participating was an initial interest in usability evaluation, based on knowledge about another company's successful experiences. Furthermore, the customer of the product in question required a usability evaluation.

Participants. Throughout the collaboration, the main contact person was the user consultant for the product in question. The user consultant was responsible for verification and quality

assessment of the product. In addition, a developer observed and provided technical assistance during the usability evaluation.

Procedure. The study was conducted in three steps. The first step was an introductory meeting with the user consultant. The purpose was to gain an overview of the product and clarify mutual expectations.

The second step was the usability evaluation. The results from the evaluation were emailed the day after the evaluation. Interviews were made shortly after. The results from the evaluation were presented along with redesign proposals at a meeting.

The third step involved two parts. Six months after the evaluation, the user consultant was interviewed to assess the effect. Two months later, the user consultant was interviewed again about the current obstacles in the company.

Table 1. Essential statements from company A and B before and after the trial evaluation.

Obstacle	Initial statements	Final statements
Resource demands	Company A: "It would be a high increase in the price and maybe delay the development two weeks or more. The customer should then be ready to pay 100.000 kr. more than now." Company B: "...when we don't know what is needed to conduct an evaluation, then it will probably take too much of our time."	Company A: "I can see it being conducted on special products or occasions, places where we deem it extra important or are suspicious about a poorly designed user interface. But nothing regularly, there is typically no time for it in our development process." Company B: "There are no resources for usability tests, we really want to, but there's no money for it at the moment."
Lack of knowledge	Company A: "Knowledge about the right solution is an obstacle to integrating usability evaluation in the development process." Company B: "...we have very little knowledge about usability evaluations."	Company A: "... the evaluations gave an insight into how the system was actually used by a prospective end user." Company B: "I have gained some knowledge, but not enough to conduct an evaluation on my own."
User involve-ment	Company A: "The users don't think enough about what they are shown. If they see something smart, they want it. They don't think about the problems a new solution can generate."	Company A: "The usability problems occurred unexpectedly, and related more to user errors or lack of users' understanding."
Structure of the system	Company A: "Often, the database layer and other function-related layers are limiting the user interface. You lock a lot in the beginning of the project."	Company A: "...the development system and environment is not up to date."
Manage-ment interests		Company A: "I actually don't think the need for usability evaluations is apparent to upper management. Usability is taken for granted..."

Setting. Most meetings were held at company B. The post-evaluation meeting was held at the university, and the evaluation was conducted in our usability laboratory.

Data collection and analysis. The interviews with the user consultant were video recorded. Each interview was based on an interview guide [15]. The recordings were processed with "condensation of meaning" [15]. The analysis was done in exactly the same way as with company A.

4. Results

This section presents the results of our study in the two software organizations. The results are summarized in Table 1.

4.1 Resource Demands

The two software organizations initially had some obstacles in common. Both were convinced that usability evaluation was very time consuming and costly, as stated by the section manager in company A. The developer and user consultant also agreed that time and money were major obstacles. The main obstacle was the expectation about the time it would take to conduct the evaluations and make software changes.

Company B was looking for an inexpensive opportunity to evaluate the usability of their product. The resource demands of usability evaluation were underlined by the user consultant from company B in the following way; "The resource demand will always be an obstacle" and "... when we don't know what is needed to conduct an evaluation, it will probably take too much of our time".

In the final meetings, both organizations still stated resources in relation to time and money as being a main obstacle. It was most prominent in company B, where the user consultant made statements such as "We don't have the resources to conduct a usability test." and "... it would take too much time for us ... we don't have the experience".

Company A expressed this obstacle both in the interviews and the forms. In a discussion of gains from usability evaluations, the user consultant said "... it would be too expensive to reveal the problems this way". When asked about the downsides of usability evaluation, the developer stated "I still think a lot of time is spent on it. You really don't have much time here". The user consultant stressed that resources is the most important factor "It all comes down to resources; the bottom line is always the focus point."

Resource demand as a main obstacle was also apparent in the forms. The section manager did only consider it relevant for special cases. On the other hand, he was surprised by the prompt delivery of results, and the user consultant concurred "The results were delivered very fast. I assumed it would take 3-4 weeks."

The resource demand of introducing usability evaluation was initially one of the major obstacles for both companies. The use of the low-cost method [14], gave the user consultant from company A an entirely different view "It changed my idea of how much time usability evaluations take." The section manager's attitude also changed. The change in company B was even more prominent as the user consultant expressed "If there is money for usability evaluation, we will certainly deploy it in the development process".

4.2 Lack of Knowledge

Both companies stated that their knowledge of usability evaluation was initially at a very low level. Company B had some knowledge from another software organization that conducted usability evaluations, but only on the general level that usability evaluation can give useful information to developers. They did not have any knowledge about usability work practices. Company A had some knowledge from another department, where a usability evaluation had been conducted once, but no evaluations had ever been done in the care management unit. The lack of knowledge also extended to the users' application of the system as the section manager stated "It would be great to get the knowledge into the organization; this could be used by the developers to make the product more usable for the end user." The developer agreed; "We lack knowledge about the users' professional world."

Lack of knowledge about usability evaluations was still expressed as an obstacle for both companies after the demonstration of usability evaluation. For company A, this applied to knowledge about evaluations and usability in general. The developer stated "As a developer, I find it hard to decide when to involve users in the development process." In relation to the question when users should be involved, the section manager said "Usability evaluations can only be conducted in the final phases of a development process." The lack of knowledge about usability evaluation was also expressed by the user consultant from company B "I have gained some knowledge, but not enough to conduct an evaluation on my own."

The lack of knowledge regarding the users' application of the system as well as usability evaluation in general was the obstacle that was affected most in our study. An example of this was given by the user consultant in company A "...three of us discussed a design solution, but we were not able to agree, so we called a user and found the answer ... if you want something tested, you can just grab a user and ask for his or her opinion." This approach had not been employed prior to our demonstration of usability evaluation. The demonstration made the employees experience that users can be involved in a constructive way in the development process. Other statements from the user consultant in company A underlined that the usability evaluation gave insight into the users' work routines "Your tests show that it has a lot to do with work routines, and that has given us motivation for following up in the next release." The importance of the evaluators was also stressed "Your tests show some subconscious things, and the users don't catch them themselves. There has to be an observer to catch those things."

In the post-evaluation meeting and the final meeting with company B, several findings pertaining to the lack knowledge were emphasized. The user consultant and developer expressed a general satisfaction with the evaluation. Observing all sessions as they happened, gave them "… an insight into the way the system was actually used by a prospective end user", as expressed by the user consultant. The evaluations revealed problems that had not previously been identified by the user consultant or developer. Both of them agreed upon the usefulness of this insight and thereby of the evaluations. In the final meeting with the user consultant, these attitudes and viewpoints were still completely intact. She said "When our new product is almost finished, it will be evaluated in the same manner ..."

The insights gained from the usability evaluations were also mentioned in the final meeting with the user consultant "You can tell if the system is intuitive to use, if they can push the right buttons and read the display. These are things we cannot answer by discussing it in the development department. It is things we don't think about." The user consultant also stated that the results from the evaluation were of great use in her daily work. In certain design discussions, she was able to use the results as examples of actual user behaviour. The introduction also had an impact on the user consultant's knowledge about usability evaluations. Initially, she had no knowledge about it, but in the final meeting she mentioned; "If we need a test of a future product, we know what usability evaluation is and what it can be used for, and we know when to test. So we can use this process for a lot of purposes."

4.3 User Involvement

The two software organizations differed considerably in their thinking about end users. Company B wanted the end users to be able to use the product with a minimum of training and a very small and easy to read manual. In company A, the user consultant expressed "Our system is so complex that training is a necessity; in no way would the end user ever be able to use the system without the training we give them."

The users were contributing with proposals for changes to the system developed by company A, but this was actually considered more of a complication. For example, the developer mentioned "The users lack knowledge about the development process and the time it takes", and the user consultant stated "The users do not have an overview of the system and its structure, and they might disagree about new functionality." The section manager also mentioned difficulties related to the involvement of the end users "The users don't think enough about what they're shown. If they see something smart, they want it. They don't think about the problems a feature can generate."

After the demonstration, obstacles relating to user involvement were only expressed in company A. The user consultant spoke of their users as being too numerous and geographically spread "... to reach out to 50% or even 10% of our users, that cannot be done. We have too many users." Furthermore, usability evaluation of a product during development would be hard to conduct, because they would be forced to use inexperienced users, which would make the tests difficult "...it would most likely "drown" in explanations of the new functions." The section manager expressed a similar concern about involving users in an evaluation "For the users to be involved in a test, they would have to be pulled away from their work. That costs money for the customer and will be a burden." Company A was also reluctant to involve users, because their understanding of the problems found in the usability evaluation was that it was the users' lack of knowledge about the system that caused the problems, as expressed by the developer "The usability problems occurred unexpectedly, and related more to user errors or lack of users' understanding.".

The introduction of usability evaluation gave the participants from company A a deeper insight into the users' way of using the system. Yet this insight also emphasized user involvement as an obstacle. For example, the user consultant expressed it this way "Are the problems occurring just because the users have adopted a wrong work routine ... the users lack an understanding of the use of the system."

4.4 Structure of the System

Company A had an obstacle regarding the structure of the system. This was expressed by the section manager. The developer also mentioned the difficulties with the system structure "The system is used in different ways. With major changes there is a risk of removing existing functionality and introducing new errors in properly working parts of the system." Although the structure in itself is not an obstacle to usability evaluation, correcting the problems found could be very difficult as expressed by the user consultant "Some parts of the system are hardcoded and cannot be changed, although the users see it as a small change."

The introduction of usability evaluation had no tangible effect on this obstacle, but reveals a need to prepare developers for potential changes in the system structure.

4.5 Management Interests

The participants from company A expressed an obstacle in relation to management, but only after the demonstration. When asked how apparent the importance of usability was for management, the developer said "I actually don't think it's apparent for management. Usability is taken for granted ..." The user consultant stated in relation to this obstacle "My attitude and position to the matter isn't opposed to it, but reprioritization has to come from the management level." In company B, the obstacle of management interests was also expressed by the user consultant "Management has decided to postpone usability evaluations until sales have gone up."

This obstacle was not identified in the initial statements, but only in the final statements. It emerged because of our direct question whether the company would consider deploying usability evaluation in the development process in the near future.

5. Discussion

The results of this study show that specific obstacles such as the resource demands and lack of knowledge about users and usability evaluation methods have been affected. The quick feedback from the evaluation to the software organization was a significant reason why company A would consider usability evaluation in the future. The fact that the participants from the two software organizations observed one or more test sessions increased their insight into the methodology and the users' ways of using the system considerable. This was clear from the positive comments that participants from both companies made about observing the tests.

The fact that the software organizations were affected by observing the benefits of usability evaluation is a valuable contribution of this study, and should be a point of focus in further research. This is also where this study differs from related work within this area. As mentioned in section 2, many of the previous studies have focused on providing guidelines or principles for deploying usability practices. The purpose of these has been to ease the deployment of usability evaluation in the development process [3, 6]. In contrast, the purpose of our study was to let company representatives observe the benefits of usability evaluation.

An important factor when deploying usability evaluation is the motivation of the software organization. In our study we observed a different motivation between the two software organizations. Company A's motivation for participating in the experiment was curiosity about the nature of usability evaluation and its practical use. Company B had a need to gain knowledge about usability evaluation because of customer demands. This difference in motivation might have had an impact on the obstacles identified. For example, the number of obstacles identified in company B was only two, while it in company A was four before the introduction and five after. Moreover, an obstacle identified in company A related to the users and the difficulties of meeting with the users. Company B also had difficulties with creating contact with users, but it was not expressed as an obstacle. Overall, company A had a tendency to see obstacles rather than benefits of usability evaluation, which indicate a lack of motivation that makes it even more difficult to deploy usability evaluation.

To increase the motivation, a software organization needs to experience that usability evaluation can fulfill relevant needs. Company B was more willing to deploy usability evaluations than company A after the demonstration. Another factor relating to the greater effect might have been that the employees from company B observed all the sessions of the usability evaluation, whereas the employees from company A, observed only one session. The experiences with company A also showed that decisions to integrate and prioritize evaluations had to come from top level management. Therefore it could be beneficial to include participants from that level in a demonstration.

6. Conclusion

The purpose of this study was to observe how the introduction of usability evaluation affects significant obstacles to usability evaluation in software organizations. To inquire into this, a usability evaluation was demonstrated to two software organizations. This included that we conducted a usability evaluation and presented the evaluation results to the two software organizations.

The results show that the introduction of usability evaluation provided the software organizations with insight into the users' use of the system. Furthermore, they experienced that usability evaluations are not nearly as resource demanding as expected. This illustrates that the stakeholders' attitudes to these obstacles were affected. However, none of the obstacles identified in the two software organizations were completely resolved. Two of the initial obstacles, user involvement and structure of the system, were not affected by the demonstration of usability evaluationl.

This study shows that it is possible to motivate software organizations toward usability evaluation. This was achieved through the approach in which the companies' products were evaluated. This underlines the relevance of research in this topic based on other approaches than providing guidelines and principles, which has been covered to a great extent.

There are some important limitations to our study. The two software organizations were quite similar. Also, we interviewed quite few persons in these organizations. In both organizations, we focused in particular on the benefits and time taken; we did not deal explicitly with the costs for the two organizations. The main source of data was interviews combined with forms in one of the organizations. Finally, the specific method used in the evaluations might have affected the results. It would be

interesting to extend the study to more organizations and stakeholders and use different methods both for data collection and for the evaluation.

Acknowledgments. The research behind this paper was partly financed by the Danish Research Councils (grant numbers 2106-04-0022 and 2106-08-0011). We are very grateful to the two software organizations and the stakeholders that participated in the study. We would also like to thank the anonymous reviewers for their comments and advice.

References

[1] Sarah Bloomer, Rachel Croft. Pitching usability to your organization. In interactions, ACM Press, 1998, (4,6), pages 18-26. ISSN: 1072-5520.

[2] Inger Boivie, Jan Gulliksen, Bengt G¨oransson. The lonesome cowboy: A study of the usability designer role in systems development. In Interacting with Computers, Elsevier Science B.V., (18,4), 2006, pages 601-634.

[3] Jim Carter. Incorporating standards and guidelines in an approach that balances usability concerns for developers and end users. In Interacting with Computers, Elsevier Science B.V., (12,2), 1999, pages 179-206.

[4] Susan M. Dray, Clare Marie Karat. Human factors cost justification for an internal development project. In Costjustifying usability, Academic Press, Inc., 1994, pages 111-122. ISBN: 0-12-095810-4.

[5] Carola B. Fellenz. Introducing usability into smaller organizations. In interactions, ACM Press, 1997, pages 29-33. ISSN: 1072-5520.

[6] John D. Gould, Clayton Lewis. Designing for usability: key principles and what designers think. In Communications of the ACM, ACM Press, 1985, (28,3), pages 300-311.

[7] Toni Granollers. User Centred Design Process Model. Integration of Usability Engineering and Software Engineering. In Proceedings of interact 2003. Found at: http://www.griho.udl.es/publicacions/2003/Doctoral%20Consortium%20(Interact%2003).pdf, last seen April 11th 2007.

[8] Jonathan Grudin. Obstacles to user involvement in software product development, with implications for CSCW. In International Journal of Man-Machine Studies, Academic Press Ltd., 1991, (34,3), pages 435-452. ISSN: 0020-7373.

[9] Melody Y. Ivory, Marti A Hearst. The state of the art in automating usability evaluation of user interfaces. In ACM Comput. Surv., ACM Press, 2001, (33,4), pages 470-516. ISSN: 0360-0300.

[10] Claire Marie Karat. A business case approach to usability cost justification. In Cost justifying usability, Academic Press, Inc., 1994. ISBN: 0-12-095810-4.

[11] Clare Marie Karat. A Comparison of User Interface Evaluation Methods. In Usability Inspection Methods, 1994, pages 203-233. ISBN: 0-471-01877-5.

[12] Clare Marie Karat. Cost-justifying usability engineering in the software life cycle. In Handbook of Human-Computer Interaction, Elsevier Science Inc., 2nd edition, 1997, pages 767- 778. ISBN: 0444818626.

[13] Brenda Kerton. Introducing usability at London Life insurance company: a process perspective. In CHI ’97: CHI ’97 extended abstracts on Human factors in computing systems, ACM Press, 1997, pages 77-78. ISBN: 0-89791-926-2.

[14] Jesper Kjeldskov, Mikael B. Skov, Jan Stage. Instant data analysis: conducting usability evaluations in a day. In NordiCHI ’04: Proceedings of the third Nordic conference on Humancomputer interaction, ACM Press, 2004. ISBN: 1-58113-857-1.

[15] Steinar Kvale. Interview - En introduktion til det kvalitative forskningsinterview. Hans Reitzel, 2.edition, 1998. ISBN: 87-412-2816-2.

[16] Deborah J. Mayhew. Business: Strategic development of the usability engineering function. In interactions, ACM Press, 1999, (6,5), pages 27-34. ISSN: 1072-5520.

[17] Karsten Nebe, Lennart Gr¨otzbach. Aligning user centered design activities with established software development practices. In NordiCHI ’06: Proceedings of the 4th Nordic conference on Human-computer interaction, ACM Press, 2006, pages 485-486. ISBN: 1-59593-325-5.

[18] Jakob Nielsen. Why GUI panic is good panic. In interactions, ACM Press, 1994, (2,1), pages 55-58. ISSN: 1072-5520.

[19] Jerilyn Prescott, Matt Crichton. Usability testing: a quick, cheap, and effective method. In SIGUCCS ’99: Proceedings of the 27th annual ACM SIGUCCS conference on User services, ACM Press, 1999, pages 176-179. ISBN: 1-58113-144-5.

[20] S. Reed Who defines usability? You do!. In PC//Computing, (Dec), 1992, pages 220-232. ISBN: 0-12-095810-4.

[21] Stephanie Rosenbaum, Sarah Bloomer, Dave Rinehart, Janice Rohn, Ken Dye, Judee Humburg, Jakob Nielsen, DennisWixon. What makes strategic usability fail?: lessons learned from the field. In CHI ’99: CHI ’99 extended abstracts on human factors in computing systems, ACM Press, 1999, pages 93-94,. ISBN: 1-58113-158-5.

[22] Ahmed Seffah, Alina Andreevskaia. Empowering software engineers in human-centered design. In ICSE ’03: Proceedings of the 25th International Conference on Software Engineering, IEEE Computer Society, 2003, pages 653-658.

[23] Desir Sy. Bridging the communication gap in the workplace with usability engineering. In SIGDOC ’94: Proceedings of the 12th annual international conference on Systems documentation, ACM Press, 1994, pages 208-212. ISBN: 0-89791-681-6.

[24] Leslie Tudor. Obstacles to user involvement in software product development, with implications for CSCW. In Human factors: does your management hear you?, ACM Press, 1998, (5,1), pages 16-24. ISSN: 1072-5520.

User Involvement in Icelandic Software Industry

Marta Kristin Larusdottir
Reykjavik University
Kringlan 1
103 Reykjavik, Iceland
+354-599 6200
marta@ru.is

Olöf Una Haraldsdottir
Reykjavik University
Kringlan 1
103 Reykjavik, Iceland
+354-599 6200
olofh06@ru.is

Bright Agnar Mikkelsen
Reykjavik University
Kringlan 1
103 Reykjavik, Iceland
+354-599 6200
brigt06@ru.is

ABSTRACT
This paper reports the first results from a recent study done on user involvement in Icelandic software industry. A questionnaire survey was made to gather information on the software processes used and to what extent user involvement methods are used by software developers in the different processes.

The results show that the majority of the respondents use their own process where they have adjusted their development process to their needs. More than one third of the respondents use the agile process Scrum. That group is the most skeptical one when rating the importance of usability in software development. Meetings are the most popular method for involving users.

Categories and Subject Descriptors
H.5.2[User Interfaces] User-centered design, Theory and methods.

Keywords
Software processes, User involvement methods, User centered software development.

1. INTRODUCTION
A numbers of studies have been done in different countries to gather information on how practitioners use methods for involving users in the software development, e. g. [1, 2, 4, 5, 7, 9]. When the results from these studies are compared, it can be seen that the emphasis in one country can differ to some extent to the emphasis in another country regarding user involvement methods used and how the respondents rate the methods. A study like this has not been done in Iceland so far.

The agile software development process has been growing in popularity in Iceland for the last five years or so, where the Scrum process has been the most popular one. In Scrum the projects are split up in two to four weeks long iterations called sprints, each ending up with a potential shippable product. Scrum heavily emphasizes on self organizing and well compounded teams, typically with 6 – 8 interdisciplinary team members [6]. Traditional Scrum has been criticized for not involving users in their software process and for not adequately address their usability needs, for example in [8].

In this paper the following research questions are analyzed:

- What software processes are used in the Icelandic software industry today?
- How do software developers rate the importance of usability? Is there some variance according to the process used?
- Which methods do software developers use to involve users in the software development? Is there some variance according to the process used?

2. MATERIALS AND METHOD
An online questionnaire was created in the QuestionPro tool for gathering data on the research questions. The target respondents were software developers in Iceland. There are not that many specialists in Human-Computer Interaction in Iceland so the software developers are the ones that contact users during the software development and they should have had one or two course in their education for learning methods to involve users.

The survey was sent out to two mailing lists, one containing 100 members called the Agile-group and the other containing approximately 100 women in information technology (IT-women). The survey was also posted on Facebook within a group of the Computer Scientists Association containing 256 members. It is possible that the mailing lists and the group overlap and therefore we estimate that the survey reached approximately 300 target respondents.

According to the Federation of Icelandic Industries [3] there were 2.071 jobs in the Icelandic Software Industry in the year 2004. It is hard to say what the number is now because between 2004 and 2008 there was a big growth in the field but in October 2008 the financial crisis changed the picture a lot. Still the software industry has not been as much affected as other industries, so we estimate that there were around 2.000 employees working in the software industry at the time of the survey.

The number of respondents was 82 so we estimate that around 25% of the people contacted did respond. The majority of the respondents 93% had B. Sc. degree or M. Sc. Degree in either Computer Science or Engineering. More than half of the respondents or 54% had 10 years experience or more in the software industry. More than half of the respondents 56% were male and 44% women. According to the Federation of Icelandic Industries [3], 24% of the employees in software industry in Iceland were women in 2004, so our sample is biased towards women.

Right now we are analyzing the data, so this paper describes the first results from the survey.

Interact 2009, August 24–28, 2009, Uppsala, Sweden.

3. RESULTS

In the following answers to the three research questions will be described.

3.1 The software processes used

When asked about the process that the developers use for software development, 44% of the respondents say that they use their own process, where they have probably adjusted some known process to their needs. Furthermore 37% use Scrum, which has grown in popularity the last five years or so in Iceland. The remaining 19% use other processes, including for example the Waterfall process and Extreme programming.

3.2 The importance of usability

When asked to rate the importance of usability the definition of usability was first described to them in the following way: "Usability is a qualitative attribute that assesses how easy user interfaces are to use. Usability is mainly made up of three factors: Effectiveness – Can the users solve their tasks with the software? Efficiency – Can the user solve their tasks without major problems? Satisfaction – How satisfied are the users?" The respondents were asked to answer if they agreed or disagreed to the statement that usability is important for the success of the software. The developers that used Scrum were the most skeptical, as can be seen on Figure 1. Twelve percent of the respondents that use the Scrum process answer that usability is neither important nor unimportant. Sixty one (61%) said they strongly agree, but 72% of those that use their own process said they strongly agreed to the statement. One explanation could be that the Scrum process is primarily used in some industrial sectors where usability indeed is not such important. Further analysis of the data is needed to check that.

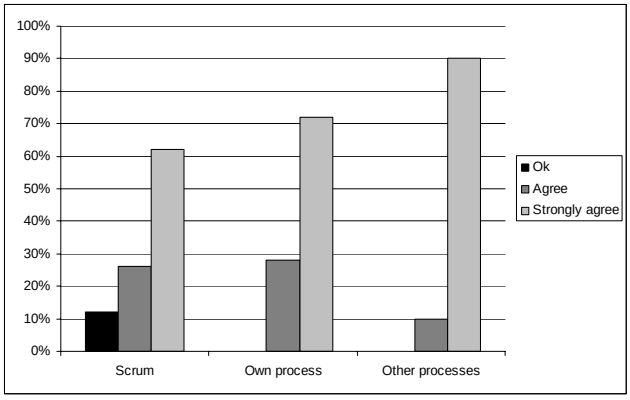

Figure 1: Usability is important

3.3 User involvement methods used

When asked what user involvement methods the developers had used the results show that some of the methods are used in all processes but for other methods there is bigger variance. Meeting with users are very commonly used in all processes but questionnaires and guidelines are not much used. It is a rather positive result for the participants of this workshop that the Think-aloud method is used by around half of the respondents and the participants using Scrum are the ones that have the highest number of usage of the Think-aloud method.

Table 1: The User Involvement Methods used in each software development process

	Scrum	Own	Other
Questionnaire or Survey	33%	27%	0%
Interviews	44%	62%	86%
Personas	33%	46%	29%
Scenarios	56%	46%	86%
Use Cases	67%	46%	86%
User Stories	78%	42%	71%
Meetings	**89%**	**96%**	**100%**
Guidelines	28%	35%	29%
Paper Prototypes	56%	35%	71%
Digital Prototypes	67%	50%	71%
User tests (Think aloud)	72%	54%	43%
Other methods	17%	12%	29%

4. CONCLUSIONS

One third of the Icelandic developers use the Scrum process and that group does not rate usability as highly as developers using other processes. We do not have any results explaining this yet, but this is really worth looking at in future work. When looking at what user involvement methods are used in each development process this trend is not that obvious. The surprising result there is that the most popular method is meetings with users even though that has not been taught in any text books on user involvement.

5. REFERENCES

[1] Bygstad, B., Ghinea, G., & Brevik, E. (2008). Software development methods and usability: Perspectives from a survey in the software industry in Norway. *Interacting with computers* , 375-385.

[2] Gulliksen, J., Boivie, I., Persson, J., Hector, A., Herulf, L. (2004). Making a difference: a survey of the usability profession in Sweden. *Proc. of NordiCHI 2004,* ACM Press (2004), 207-215.

[3] IT and Communication Technology in Iceland. (n.d.). Retrieved 15th of May, 2009, from Ice Trade Directory: http://www.icetradedirectory.com/icelandexport2/english/industry_sectors_in_iceland/it_and_communication_technology_in_iceland/

[4] K. Vredenburg, Mao, J. Y., Smith, P. W., Carey, T. (2002) A Survey of User-Centered Design Practice. *Proc. CHI 2000,* CHI Letters 4(1), 471-478.

[5] Rosenbaum, S, Rohn, J. A., Humburg, J. A. (2000). Toolkit for Strategic Usability: Results from Workshops, Panels, and Surveys. *Proc. CHI 2000,* CHI Letters 2(1), 337-344.

[6] Schwaber, K. (1995). Scrum development process. *OOPSLA'95 Worshop on Business Object Design and Implementation.*

[7] Seffah, A., Metzker, E. (2004). The Obstacles and Myths of Usability and Software Engineering, Communications of the ACM (2004), 47(12), 71-76.

[8] Singh, M. (2008). U-SCRUM: An Agile Methodology for Promoting Usability. *AGILE '08. Conference,* (pp. 555-560).

[9] Venturi, G., Troost, J. (2004). Survey on the UCD integration in the industry. *Proc. NordiCHI 2004,* ACM press (2004), 449-452.

Early user-testing before programming improves software quality

John Sören Pettersson
Department of Information Systems
Karlstad University
Karlstad, Sweden
+4654 700 2553
John_Soren.Pettersson@kau.se

Jenny Nilsson
Department of Information Systems
Karlstad University
Karlstad, Sweden
+4654 700 1135
Jenny.Nilsson@kau.se

ABSTRACT
This position statement does not focus on usability although it presents data from a software up-date cycle where several usability- and user-centred methods were used. The important lesson learnt is that a better (more complete) specification before programming results in fewer errors in the code and that such a specification can be reached by user tests of interactive mockups.

Categories and Subject Descriptors
D.2.1 [**Software Engineering**]: Requirements/Specifications – *Elicitation methods.*

D.2.2 [**Software Engineering**]: Design Tools and Techniques – *Evolutionary prototyping, User interfaces.*

General Terms
Design, Experimentation, Human Factors.

Keywords
Software quality, Early user-testing, Wizard-of-Oz prototyping.

1. CASE STUDY
Frequent testing of developing software can certainly increase the usability in the program. However, as we found in a case study, the method seems to continuously introduce changed or new requirements which in turn results in more complex code and thereby more errors. This case study consisted of a large update cycle of a software package in the area of decision support system for civil protection. The update involved a complete re-programming of the four largest modules. Several smaller updates had been made prior to the large update cycle, and requirements for the update had (as always) been collected from the large user groups. The organisation had routines for collecting requirements from users, client organisations, and other stakeholders.

There was thus much resemblance of their approach to principles found in user-centric approaches such as the MUST method [2]. The organisation had however recognised that usability was an issue even if the type of functions provided by the system was requested by client organisations and their employees. They had also included a continuous process of debugging using experienced users and content experts in their update cycles. One can say that the developers were not aware of the methodological critique expressed in one paper as "Close Co-operation with the Customer Does Not Equal Good Usability" [4] (cf. also [1]). Through an HCI student's exam work for the organisation, its developers became aware of the Wizard-of-Oz method by which one can test mocked up designs as if they were already implemented [3]. A more experienced Wizard (second author) was hired as a usability expert and design-aide and stayed through the 3-year update project of the software package.

Due to the size of the project, the Wizard could not pre-test every module: one of the four largest modules was not mocked up in advanced. Figure 1 shows the two user-centred processes employed in this large update project (the debugging commenced half a year after programming had started).

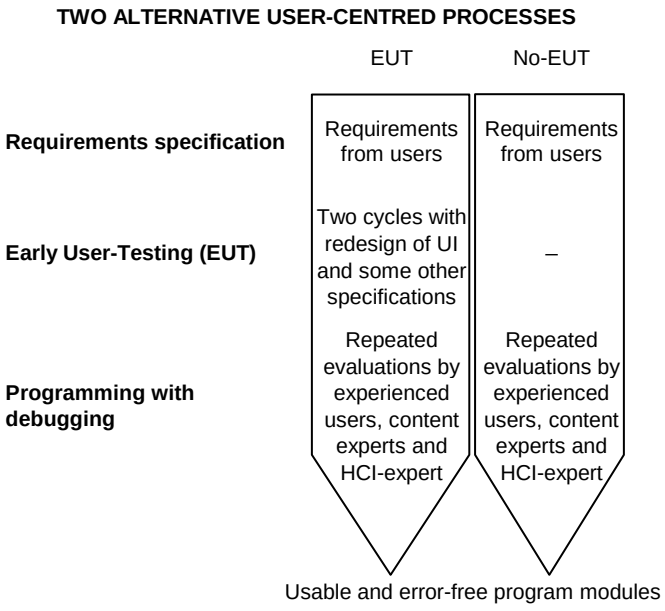

Figure 1. Flow of work with and without Early User-Testing

2. ERROR RATES
The debugging process showed an interesting difference in the number of errors found in the module lacking pre-testing and a

Conference'04, Month 1–2, 2004, City, State, Country.

Table 1. Error rates relative to program size (MB and # of files)

Error type / Program	Prio 1		Prio 2 + 3		Priority 1,2,3	
	# / MB	# / files	# / MB	# / files	# / MB	# / files
Early User-tested module of 1.5 MB and 145 files	4.67	0.05	68.00	0.70	72.67	0.75
Not-EUT module of 2.0 MB and 230 files	32.50	0.28	101.00	0.88	133.50	1.16
Error rates proportionally (EUT / Not-EUT)	*0.14*	*0.17*	*0.67*	*0.80*	*0.54*	*0.65*

Note: Priority 3 was in the error reports noted as "Next version", often new requirements, while Priority 1 was "critical errors".

comparable pre-tested module. Table 1 indicates both the size (in MB) and the number of files of the two modules. Error rates are given both in relation to size and number of files. The EUT-developed module has about one-fifth of the error rate of the not-EUT module for the "Prio 1" errors (called "critical errors" in the debugging reports). In total, the error rate for the first module is only half of what was found in the second module.

It is not meaningful to compare program modules without considering the relative complexity of each module. The two other EUT-modules were only half the size of the one we select for this error comparison but contained, relative to their size, many more errors than the modules in Table 1. However, these other modules contained specific, database-related complexities and can only be used for certain comparisons (2.2).

2.1 The debugging process

The debugging process commenced nearly a year before the final launch of the new version. The debugging was conducted by three groups which were very familiar with the functional requirements: a group of very experienced users, the HCI expert, and the content managers for the different modules' databases.

The bug-finding by experienced users sometimes resulted in new requirements coming up. Interestingly, this was also the case for the debugging made by the content experts (who had not been involved in the pre-tests before programming; they had only seen and accepted the requirements specifications).

2.2 New requirements

For the first module in Table 1 there was only 4 new requirements coming up in the extensive debugging process while for the second module there was 13. This we hold to be the source of many of the other errors. When new functions are introduced into the developing process, it is harder for the programmers to maintain a clean and easily predictable code.

That early user-testing can capture many requirements was shown by a third module, smaller in size than the two modules in Table 1 (0.7 MB and consisting of only 55 files). This third module mainly consisted of a library and the content expert of this module found many faults during the debugging process: among these were in effect 24 new requirements. In the HCI expert's (i.e. Wizard's) opinion, most of the new requirements would have been possible to spot if the content expert had been included in the pre-testing, which could have been done *without the wizard setting up special test scenarios for content experts*. This is important when the Wizard-of-Oz method is used as the method incurs some extra costs when mockups have to be prepared before tests.

3. EARLY USER-TESTING

The much criticized Waterfall model for systems development, where all specifications should be settled before the laborious tasks of modelling and programming take place, admittedly has some advantages, but *only if all requirements really can be settled in advanced*. By early prototyping designers can approach this goal. In the case study, the Wizard-of-Oz method was used with user interfaces often based on previous versions of the system. What was needed was elaboration of the interaction design and uncovering interdependences between various function requirements. This was met by the WOz prototyping, which was conducted in two rounds: a first one on a rough design with 8 participants; a second one six months later on a detailed design with 5 participants. Although the interaction is 'real' in WOz experiments, the graphics can be crude in early design phases.

Setting up a WOz environment for testing is laborious as the Wizard must have control over what the user sees on the monitor (and hears from the loudspeakers), but in our research group we have developed a 'general-purpose' WOz system which we call Ozlab ,which facilitates the setting up of tests enormously (cf. e.g. [5]). A WOz set-up also allows designers to probe their own designs and find interaction bugs even before testing.

Still to evaluate is how much more costs the error-correction took in comparison with the cost for the Wizard work, but from our experiences of this project (and noting the difference in salaries between usability people and programmers…) it seems a safe bet that the EUT injected as in Figure 1 pays of very well to say nothing of how much frustration is saves.

4. REFERENCES

[1] Ambler, S.W. 2004. Tailoring Usability into Agile Software Development Projects. Maturing Usability, eds. Law, Hvannberg & Cockton. Pp 75-95. Springer-Verlag

[2] Bødker, K., Kensing, F. and Simonsen, J. 2004. Participatory IT Design. Designing for Business and Workplace Realities. MIT Press.

[3] Gould, J. D. and Lewis, C. 1985. Designing for usability: key principles and what designers think. Com. ACM 28:300-311.

[4] Jokela, T. and Abrahamsson, P. 2004. Usability Assessment of an Extreme Programming Project: Close Co-operation with the Customer Does Not Equal Good Usability. PROFES 2004 Proceedings, pp 393-407. Springer-Verlag.

[5] Molin, L. 2004. Wizard-of-Oz Prototyping for Cooperative Interaction Design of Graphical User Interfaces. Proceedings of the Third Nordic Conference on Human-Computer Interaction, 23-27 October, Tampere, Finland, pp. 425-428.