COST294-MAUSE Workshop

Downstream Utility: The Good, the Bad, and the Utterly Useless Usability Evaluation Feedback

Toulouse, France • November 6th 2007

Effie L-C Law, Marta Kristín Lárusdóttir and Mie Nørgaard
(Editors)

http://cost294.org/
**ACKNOWLEDGEMENT**

First of all, we are very grateful to the local organizer - IRIT Lab, University Paul Sabatier (Toulouse 3), France, especially Marco Winckler and Philippe Palanque, who have strongly supported us to hold our 4th COST294-MAUSE Open Workshop “Downstream Utility: The Good, the Bad and the Utterly Useless Usability Evaluation Feedback” (http://cost294.org/downstream). Thanks must also go to the authors of the workshop’s papers, whose contributions serve as rich sources of stimulation and inspiration to explore the issues of interest from multiple perspectives. The quality of the contributions could further be ensured and improved with the generous help of the program committee members (Table 1). Their effective and efficient review works are highly appreciated.

**Table 1**: List of the reviewers of the COST294-MAUSE workshop “Downstream Utility”, 6 Nov., 2007

<table>
<thead>
<tr>
<th>Name</th>
<th>Affiliation</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Følstad Asbjørn</td>
<td>SINTEF</td>
<td>Norway</td>
</tr>
<tr>
<td>Gilbert Cockton</td>
<td>University of Sunderland</td>
<td>UK</td>
</tr>
<tr>
<td>Kasper Hornbæk</td>
<td>University of Copenhagen</td>
<td>Denmark</td>
</tr>
<tr>
<td>Effie Law</td>
<td>ETH Zürich/University of Leicester</td>
<td>Switzerland/UK</td>
</tr>
<tr>
<td>Marta Lárusdóttir</td>
<td>Reykjavik University</td>
<td>Iceland</td>
</tr>
<tr>
<td>Rosa Lanzilotti</td>
<td>University of Bari</td>
<td>Italy</td>
</tr>
<tr>
<td>Mie Nørgaard</td>
<td>University of Copenhagen</td>
<td>Denmark</td>
</tr>
<tr>
<td>Tobias Uldall-Espersen</td>
<td>University of Copenhagen</td>
<td>Denmark</td>
</tr>
</tbody>
</table>

Last but not least, we express gratitude to our sponsor – COST (European Cooperation in the field of Scientific and Technical Research; http://cost.cordis.lu/src/home.cfm). The COST Office operated by the European Science Foundation (ESF) provides scientific, financial and administrative support to COST Actions. Specifically, the COST Action 294 (http://www.cost294.org), which is also known as MAUSE, was officially launched in January 2005. The ultimate goal of COST294-MAUSE is to bring more science to bear on Usability Evaluation Methods (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. The current Workshop is the second open workshop implemented under the auspices of COST294-MAUSE. As with other past and forthcoming events of COST294-MAUSE, we aim to provide the participants with enlightening environments to further deepen and broaden their expertise and experiences in the area of usability.
FOREWORD

Motivation and Background

**Downstream utility**, in the context of usability evaluation method (UEM), has been described as:

“…downstream utility of UEM outputs ... depends on the quality of usability problem reports ... the persuasiveness of problem reports, a measure of how many usability problems led to implemented changes ...evaluating the downstream ability of UEM outputs to suggest effective redesign solutions through usability testing of the redesigned target system interface.” (Hartson, Andre & Willinges, 2003, p.168)

“The extent to which the improved or deteriorated usability of a system can directly be attributed to fixes that are induced by the results of usability evaluations performed on the system” (Law, 2006, p.148)

These descriptions converge to a common, basic idea that **usability evaluations of a system lead to redesign proposals whose effectiveness can be evaluated by re-testing the changed system.**

Wixon’s (2003) radical claim that it is irrelevant whether a system’s total set of UPs can be uncovered, because the true goal of usability testing lies not in finding defects but in fixing them has set off a growing interest in the topic of downstream utility (cf. John and Marks [1997]). Though Cockton (2005) states that assessment of downstream utility is beyond the scope of pure evaluation methods, we argue that the critical element of downstream utility – persuasiveness – is somewhat determined by the choice of UEM and how it is executed. For instance, outcomes of heuristic evaluation presumably are less persuasive than those of user tests, and user tests performed by designers/developers themselves seem more persuasive than those performed by usability professionals. Consequently, it is meaningful to compare downstream utility of different UEMs as well as investigate how developers, designers and project managers, who are supposed to be beneficiaries of usability evaluation feedback, assess a method’s utility and how contextual factors influence such an assessment. Note that we can learn not only from stories of success but also of failures!

**Previous Work:**
The project COST294-MAUSE hosts four working groups, of which especially Group 2 focuses on comparing UEMs. The group, which is led by Gilbert Cockton, held a project-based workshop in June 2007 (Salzburg, Austria) to describe and revise coding constructs used for comparing different instances of UEMs (e.g. persuasiveness, value to development, redesign complexity). These constructs were applied to sets of usability problems (UPs) from various domains. Coding construct definitions and problem sets are accessible via our project instrument - MAUSE Wiki. Furthermore, two workshops on the interplay between usability evaluation outcomes and system redesigns were held in NordiCHI 2004 (and the resulting special issue of IJHCI 2006, Hornbæk & Stage, 2006) and CHI 2007 (Uldall-Espersen et al. 2007), and an interactive experience session was held in CHI 2007 on recommendations (Molich et al. 2007)

**Goals:**
The workshop seeks to:

1. Identify what type of information developers (and other stakeholders) find **worthwhile** or **worthless** in usability evaluation reports, and why
2. Identify which **format** of usability evaluation feedback developers (and other stakeholders) find useful and usable (e.g., video vs. written, specific ways of data clustering), and why
3. Study existing quantitative and qualitative methods to evaluate different UEMs downstream utility such as the quality of usability feedback, a UEM’s ability to generate redesign solutions, and the effectiveness of such solutions
4. Validate the scope, reliability and usability of the pre-defined coding constructs and coding scheme (i.e., Cockton & MAUSE, 2007) for evaluating downstream utility
5. Refine the notion of downstream utility in usability evaluation.
Target Groups and Submission:
Participants of this workshop include software engineers, usability researchers and practitioners, and students/academics of HCI. Each of the fourteen workshop submissions has been reviewed and is of 2-6 pages in SIGCHI format. Topics of the papers address one of the following topics:

- Case studies assessing the impact of usability evaluation feedback on system redesign in terms of quantitative measures and/or qualitative data;
- Experience reports illustrating which kinds of feedback from usability evaluation that stakeholders find particularly useful or utterly useless;
- Theoretical frameworks for analyzing different aspects of downstream utility, such as psychology of developers;

About the Workshop Proceedings
All the fourteen submissions are included in the workshop proceedings with an ISBN. Copyright remains with individual authors, who have the full right to submit the current or a modified version of their paper to other venues. We are grateful to Marco Winckler for the design of the cover of the proceedings.

References
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhancing the Downstream Utility of Usability Evaluations with Pattern-based Recommendations</td>
<td>pp. 1-4</td>
</tr>
<tr>
<td>Martin Schmettow, University of Passau, Germany</td>
<td></td>
</tr>
<tr>
<td>Usability Work: A Human Activity</td>
<td>pp. 5-7</td>
</tr>
<tr>
<td>Mie Nørgaard, University of Copenhagen, Denmark</td>
<td></td>
</tr>
<tr>
<td>Two Styles for Presenting Usability Evaluation Results</td>
<td>pp. 8-10</td>
</tr>
<tr>
<td>Timo Jokela and Niina Kantola, Oulu University, Finland</td>
<td></td>
</tr>
<tr>
<td>Usability Case – Integrating Usability Evaluations in Design</td>
<td>pp. 11-13</td>
</tr>
<tr>
<td>Marja Liinasuo and Leena Norros, VTT Technical Research Centre of Finland, Finland</td>
<td></td>
</tr>
<tr>
<td>Facilitating Usability Work Using Different Perspectives</td>
<td>pp. 14-17</td>
</tr>
<tr>
<td>Tobias Uldall-Espersen, University of Copenhagen, Denmark</td>
<td></td>
</tr>
<tr>
<td>Downstream Utility: Stepping Back</td>
<td>pp.18-20</td>
</tr>
<tr>
<td>Ann Blandford, UCLIC, United Kingdom</td>
<td></td>
</tr>
<tr>
<td>Maximising the Uptake of Usability Results: A Consultant’s View</td>
<td>pp.21-24</td>
</tr>
<tr>
<td>Nigel Bevan, Professional Usability Services, UK</td>
<td></td>
</tr>
<tr>
<td>Evaluating Procurement, Usability and Off-the-Shelf Office Software</td>
<td>pp.25-29</td>
</tr>
<tr>
<td>Asa Cajander, Elina Eriksson, and Jan Gulliksen</td>
<td></td>
</tr>
<tr>
<td>Upstream Futility Blocks Downstream Utility</td>
<td>pp.30-33</td>
</tr>
<tr>
<td>Gilbert Cockton, University of Sunderland, UK</td>
<td></td>
</tr>
<tr>
<td>Downstream Utility: Understandable or Reliable</td>
<td>pp.34-36</td>
</tr>
<tr>
<td>Alan Woolrych, University of Sunderland, UK</td>
<td></td>
</tr>
<tr>
<td>How to Convince Stakeholders to Take up Usability Evaluation Results?</td>
<td>pp.37-39</td>
</tr>
<tr>
<td>Regina Bernhaupt and Astrid Weiss, University of Salzburg, Austria</td>
<td></td>
</tr>
<tr>
<td>Exchangeable Problem Sets Described with Evaluator Defined Constructs</td>
<td>pp.40-44</td>
</tr>
<tr>
<td>Gudmundur F. Jonasson and Ebba T. Hvannberg, University of Iceland, Iceland</td>
<td></td>
</tr>
<tr>
<td>Developer Effect in Downstream Utility</td>
<td>pp. 45-49</td>
</tr>
<tr>
<td>Effie L-C Law, ETH Zürich/University of Leicester, Switzerland/UK</td>
<td></td>
</tr>
<tr>
<td>Case Study: Are CUP Attributes Useful to Developers?</td>
<td>pp.50-54</td>
</tr>
<tr>
<td>Trausti Þorgeirsson† and Marta Kristín Lárusdóttir‡,</td>
<td></td>
</tr>
<tr>
<td>†Universiy of Iceland,‡Reykjavik University, Iceland</td>
<td></td>
</tr>
</tbody>
</table>
Enhancing the Downstream Utility of Usability Evaluations with Pattern-based Recommendations

Martin Schmettow
University of Passau, Information Systems II
94032 Passau, Germany
schmettow@web.de

ABSTRACT
The position held in this paper is that usability evaluations should integrate the constructional perspective developers are familiar with. This position is discussed regarding recent research results. As a proposed solution, the Usability Pattern Inspection method is introduced as an evaluation method with a strong notion on design. Supportive empirical evidence is outlined and lessons learned are taken as directions for further improvements and research on the method.

Author Keywords
Usability Inspection, Usability Patterns, Design Recommendations, Downstream Utility

ACM Classification Keywords
H.5.2 User Interfaces (e.g. HCI): Evaluation/methodology

MOTIVATION
Most likely, every usability specialist regularly hired for usability evaluations has already encountered the following situation: A severe usability defect was doubtlessly identified and reported to the responsible developers with careful explanations of what usage problems are to be expected. An embarrassing silence, then the inescapable question: “Well, how do you want us to design this instead?” Have you been prepared to this and can you offer a sophisticated recommendation how to fix the defect? Did you acquire your recommendations from some authoritative guidelines or best practices of design? Or were they just ad hoc and in consequence easily wiped away by technical arguments? Anyway, you may now learn about an approach of bridging from usability arguments to design recommendations, which is based on the concept of usability patterns.

POSITION
The paper presents the usability pattern inspection (UPI) as a novel inspection method, which was designed to address some deficits subsumed under the term “downstream utility”. One underlying assumption is that software developers prefer to think in terms of design and technical implementations rather than users and usage. The concept of usability patterns has a strong notion on design and should therefore ease the communication of usability defects. It may also encourage developers to conduct usability evaluations themselves, instead of delegating it to external experts. In turn, considerations on users and their behavior are introduced through the back-door by usability arguments found in well-written patterns. This is in line with the recent claim from Jared Spool, that usability experts should withdraw from conducting usability evaluations, but improve techniques and teach them to software practitioners [18].

In turn, usability experts must acquire a sound understanding of how to translate their behavioral view into appropriate design features. With patterns they can rely upon a rich and broad knowledge base and will still find their accustomed usability concepts. The usability experts’ new familiarity with design arguments hopefully increases the persuasion power of defect reports and guide for more effective defect fixing.

Or, as Dray & Siegel put forth: “UCD professionals who focus on doing ‘studies’ as opposed to generating designs and products, will always be perceived as peripheral.” [4]

RELATED WORK
Effectiveness and Downstream Utility
As a result of the “Damaged Merchandise” debate [8], we now possess a canonical framework for measuring the effectiveness of inspection methods, which accounts for defect identification and proneness to false alarms [1,9]. But effective defect identification is only one of the prerequisites of an effective usability quality process – and it can easily be increased by adding more evaluators to the process. Instead, as John & Marks point out, a defect report is only fully effective, if it convinces the development team for a change and guides an appropriate redesign [13]. The alarming results from their study, tracking the fate of defect reports, were that any of these three steps – identify, convince and guide redesign – has a loss rate of about 50%: From 56 existing defects only 6 were finally adequately fixed and, even worse, some new had been introduced. Accordingly, term downstream utility enhances the view on evaluation effectiveness beyond the mere identification accuracy to the (desperate or fortunate) fate of defect reports later in the software development process.

One major obstacle for downstream utility is the perspective gap between usability experts and software developers.
Usability experts often regard themselves as the advocates of the user and, accordingly, their favorite concepts to think and talk about are users’ goals, tasks and observed (or predicted) behavior [11]. In contrast, software developers often adhere to think in terms of features, design problems and technical solutions. This has recently been approved by a case study comparing the impact of different artifacts on the stakeholders’ perspective: "... system developers and system administrators took a constructional perspective, rather than a practical, social, aesthetic or ethical perspective, regardless of prototyping technique" [12].

But there is hope, when evaluation reports come to acknowledge and integrate both views – the behavioral and the constructional: This has been shown by Kasper Hornbæk & Erik Frøkjær: They demonstrated that design recommendations are not only appreciated for their guidance but also have considerable convincing power [10] on developers. Thus, strengthening the notion of design in evaluation methods could kill two birds with one stone.

Unfortunately, there is still a lack of approaches to systematically improve design recommendations in usability evaluations. As concluded by Molich et al., the didactic literature disregards this topic, whereas a few studies exist, including their own, which show a generally poor quality of design recommendations [14]. Besides, Dumas et al. emphasize the importance of positive feedback for compelling evaluation reports, but observe a reluctance of evaluators to do so. Again, this aspect is not addressed in common methods [5].

Recently, Jared Spool provoked with the claim, that usability specialists should completely stop giving design recommendations [18]. Slightly contradicting himself, at his keynote on the British HCI 2007 conference he called usability researchers to strive for cataloging established design solutions for common usability problems and stressed the approach of usability patterns as a promising base. This made the author feel confident about his own approach of employing usability patterns for usability evaluations.

Usability Patterns

Usability patterns are an emerging concept in Usability Engineering, which has already made it into acknowledged HCI textbooks [3]. Patterns are usually compiled by experienced practitioners, whereas the primary focus is more on internal validity instead of empirical validation, as researchers may demand. An outstanding source of high quality patterns are the Pattern Languages of Programs (PLoP) series of conferences, which employ an elaborated supervised writing process. From personal experience of the author [16], the quality of patterns is achieved mainly by the following principles: (1) A pattern (or collection thereof) should reflect the targeted domain as a non-trivial problem space with conflicting forces. This has the notion of many design problems being wicked problems, with a maximal solution not possible, but instead a good solution strives for balancing the forces. In common pattern templates this is expressed in the corresponding sections Forces and Resulting Context. (2) Further on, pattern authors are encouraged to connect their patterns to coherent pattern languages and draw references to other pattern collections. This may results in a largely complete and coherent body of knowledge, which bridges domains and covers several levels of abstraction. (3) A multiple review process assures preciseness and comprehensibility of instructions by employing practices from poetry discourse.

Whereas a large number of pattern collections exists today (see [6] for a meta collection), there have only been few approaches to utilize usability patterns systematically: Cowley & Wesson found patterns perceived as more useful than guidelines in evaluation, design and redesign tasks [2]. Finlay et al. successfully applied patterns to facilitate participatory design [7].

What makes usability patterns special is that they argue for usability (analyzing users’ needs and potential usage problems), but also describe established technical solutions. This makes the pattern approach a good candidate for bridging the usability specialists focus on users’ needs and behavior and the developers focus on technical solutions.

USABILITY PATTERN INSPECTION

The Usability Pattern Inspection (UPI) introduces some novel concepts to usability evaluations which may narrow the gap to common software development activities [15]. The UPI was explicitly designed for effective defect identification by practitioners and support for detailed redesign suggestions with patterns employed for both purposes. First, patterns help to identify defects in an interface. This is similar to comparing a UI to a standard in that a particular pattern is treated as an authoritative rule how to solve a particular design problem right. If the solution chosen for the actual design is different to the pattern or incomplete, this is regarded as a potential usability defect.

Second, and the emphasis in this paper, is how patterns help to advise design alternatives. In very brief, the same pattern, which led to a defect identification is exploited for giving the recommendation. In the simplest case, the evaluator just provides a reference to the pattern and lets the developers interpret it to the current design problem.

The UPI implements this with a structured reporting format. Every defect is reported together with the full path of how it was encountered (Task, UI location, user activity). Here, it was consciously chosen to report all patterns that were identified as matching to the design problems found. This leads to defect reports, in case a feature is questioned by a pattern, and to affirmative reports, in case a solution is supported by a pattern. Defect reports are complemented by a comment, where the inspector may describe predicted usage problems and how the pattern can mitigate or eliminate the problem. Affirmative reports have at least two advantages: First, stating positive findings appears as good feedback and can help to lower resistance of the developers [5]. Second, the tracking study mentioned above has also uncovered a number of defects that are newly introduced during redesign [13]. Testing good solutions with the UPI may act as a shield to preserve, what was already well designed.
EMPIRICAL RESULTS
The development of the UPI and research thereof is still ongoing, but some promising results exist already. In the following the results from an experimental comparison and a case study are briefly reported. Instead, the focus is on qualitative observations from these two studies in order to derive future modifications and some research questions.

Effectiveness: Experimental Results
Still, the primary purpose of any inspection method is to identify defects effectively. In this discipline the UPI has been challenged according to the performance criteria of thoroughness and validity. These measures were affirmed by a subsequent falsification test study. Finally, the UPI was found to catch up with the popular Heuristic Evaluation (HE) [17]. Another interesting finding was that mixing HE and UPI in one process yields a broader detection profile.

Downstream Utility: Experimental Results
Apparently, the first to be shown is that, indeed, usability inspectors produce more design recommendations with the UPI than with another established inspection method. This was investigated in the comparison experiment depicted above. Indeed, the UPI group produced more design recommendations than the HE group. But, this experiment also uncovered some drawbacks of the UPI: Too many defects were only reported with statements like “Pattern xy not implemented” with likewise trivial recommendations “Implement the pattern xy!”. Also, fewer predictions of usage problems were given with the UPI. It is to note that these were valid defect identifications and that these brief references are not simply useless. Still, they are better than typical statements from heuristics like “Make the feature xy easier to learn” or from user observations like “The user couldn’t figure out how to achieve xy”. Nevertheless, it is preferable that the inspectors reason about the relation of usage problems and design features. Also note, that time restrictions in the experiment were very tight. This may have engaged most participants in finding as many defects as possible and disregard verbose reporting.

Downstream Utility: Findings from a Case Study
More design recommendations does not guarantee their particular value in the redesign process. To make that true, they have to be convincing, add information to development process and be technically feasible. This was investigated in a case study at a small company with specialization on mobile solutions for nomadic workers. This company was right at the time planning a major redesign of a mobile application for facility management. Accordingly, the product manager was already sensitized for flaws and possible solutions regarding the user interface. In this situation the net value of design recommendations from a UPI session is validly assessable.

From a single UPI session, conducted by a pair of usability experts, 21 defect reports were presented to the product manager. The perceived value of the reports was assessed in a subsequent half-structured interview. In more detail, the questions were whether each report was convincing, to what extent it was previously known, and whether the recommendation was regarded as valid and technically feasible. The product manager was convinced of 16 reports to be severe (76%) and only 2 have been previously known to full extent. From the 16 acknowledged reports 12 recommendations were regarded as valid and technically feasible (75%). If the desperate 50% loss rate per step of the John & Marks tracking study was taken as a benchmark (which is a little daring), these results appear quite promising.

There also were lessons learned from the case study: In the informal part of the interview the product manager had doubts whether the UPI had uncovered all of the severe defects. Instead, he suspected that several misconceptions regarding user goals and tasks were not caught. This is not too surprising: The UPI acts primarily on the “lower” level of interaction and data presentation. It is unlikely that with the UPI an expert finds misconceptions that happened during the requirements phase of the development process, when users’ goals and expectations are captured – unless he/she is also a domain expert. The opposite problem may happen when strongly behavioral methods (e.g. usability testing in the wild) are used for informing a superficial redesign. These may produce recommendations that make large reconstructions of the user interface necessary, which is beyond the scope of the current development phase and thus useless. This may turn out as a new aspect of downstream utility: Recommendations from evaluation reports are only fully effective if they fit the current attendance to change the product.

POSSIBLE SOLUTIONS AND FURTHER RESEARCH
The empirical studies so far provide some general evidence that the UPI has the potential to become a mature technique for industrial applications. However, there are still some drawbacks to be resolved. It was shown that the UPI is superior with respect to the number of design recommendations. Given by experts in the case study setting, these were also of considerable quality and utility. In contrast, the comments in the experiment were rather poor. This will be improved by a revised reporting scheme, which now requires the inspector to argue about the predicted usage problem and how the problem is specifically solved. This will be improved by a revised reporting scheme, which now requires the inspector to argue about the predicted usage problem and how the problem is specifically solved by applying the pattern. Regardless of the particular evaluation method, the author considers a usability defect report as perfect, that includes:

- a description of the likely usage problems
- the cause for the usage problems in the current design
- a reference to an authoritative design alternative
- an argument how the alternative design mitigates or eliminates the problem
- a contextualization of the design alternative

Accordingly, the UPI training material is currently being amended by best-practice examples. These improvements also put further emphasis on taking the user perspective and hinder inspectors to work mechanically through the inspection – making them think instead. The revised UPI will undergo an experimental evaluation right this winter.
An issue for future research is to question whether evaluation methods are swiss knives fitting all purposes. As a common example, usability testing is usually preferred to inspections when usability is mission-critical and costly redesigns are easily out-weighted by future gains. Another aspect is the depth of problem identifications and recommendations which should fit the development phase. Recently, Vilbergsdóttir et al. introduced a novel reporting format based on the Classification of Usability Problems (CUP), which lets the evaluators guess the development phase affected by a problem [19]. This is an initial step, but further research into the qualitative differences of evaluation methods may improve evaluation processes and better align them to development goals.

As a final remark, the UPI is not meant as an all-or-nothing approach. Especially, the individual concepts of patterns for recommendations and positive feedback may also be combined with usability testing, in favor of most effective defect detection.

REFERENCES


11. JAN GULLIKSEN, BENGT GÖRANSSON, INGER BOIVIE, STEFAN BLOMKVIST, JENNY PERSSON, AND ÅSA CAJANDER. Key principles for user-centred systems design. Behaviour and Information Technology 22, 6 (2003), 397–409.


Usability work: A Human Activity

Mie Nørgaard
Department of Computer Science
University of Copenhagen
mien@diku.dk

ABSTRACT
Much work on usability has a clear human perspective, such as making usability results more useful for developers. Yet, most work end up detaching usability work from human activities in its aim to isolate specific phenomena important to the quality and impact of evaluation results. This paper argues that researchers and practitioners could gain from understanding usability as a human activity involving, for example, learning about and understanding usability issues, and collaborating to improve usability.

Author Keywords
Downstream utility, human, collaboration, learning, usability evaluation methods, feedback, usability report.

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation (e.g., HCI): User Interfaces—Evaluation/Methodology; D.2.2 Software Engineering: Design Tools and Techniques

INTRODUCTION
In the early seventies prominent researchers such as Naur [12] (for the English translation see [13]), and later Boehm [3] emphasized the importance of the human factor when understanding work on computer systems. These thoughts have also influenced usability work. Still, much of the work on usability concern methods, procedures, or how to report problems in a way that leads to most fixes. Of course, work on how to describe and present results from usability evaluations implicitly concern humans, namely the receivers. Yet, work that aim to isolate important phenomena for example in the use of usability evaluation methods (UEMs), seem often to view usability detached from human activities such as learning and collaborating.

To improve the downstream utility of usability work, researchers and practitioners might find it useful to understand usability in terms of activities such as learning about and understanding usability issues, and collaborating with other stakeholders to improve usability. Indeed, borrowing the terms of Naur, we might gain from understanding usability work as a human activity [13], meaning that usability work must not solely be understood as the development and use of certain evaluation methods, but also in terms of creativity, personality, individual professional goals, learning, and collaboration, to mention a few relevant aspects.

ASPECTS OF USABILITY
Literature on usability views usability work and results from a diversity of angles. Traditionally, work on usability has understood the results of usability evaluations as a presentation of problems that is quick and easy to use, see for example [5,16]. More recent, persuasiveness has been mentioned as a key factor for usability’s value and impact, and it has been coined by terms such as relevance, salience, reliability and quantity [11]. Persuasiveness has also been discussed by Nørgaard and Høegh [14] in terms of argumentation theory. Other work on the presentation of usability results concerns for example the value of redesign proposals [10], or prioritized results [9].

Furniss et al. [6] relate themes from Resilience Engineering to usability work, and argue that usability is an activity that needs to be adjustable and flexible, as when usability work adjusts to changing contextual factors. The same view has been touched upon by Nørgaard and Hornbæk [15], who report that usability experts feel the need to adjust their usability testing to changing conditions. These views relate usability work to the organisation in which it takes place, and suggests that usability is an activity that must be understood together with other activities and contextual factors, and not as a stand-alone activity that can be studied out of context.

The work discussed above, which is mainly related to methods, reporting styles, or to organisational factors, clearly has a human perspective. For example, the attempt to produce redesign proposals that can inspire developers, shows concern for the humans who receive the evaluation results. Still, it does not cover usability as a human activity that is dependent on how humans learn and collaborate.

Let us next investigate how the role of the human is discussed in usability literature.

USABILITY AS A HUMAN ACTIVITY
Dumas [16] has argued that the relationship between developer and usability expert might be the most important factor for usability’s success, more important than, for example, how usability results are fed back to developers. Others have made similar observations on the importance of human relationships, such as the relationship between usability expert and customer, users and stakeholders, and so on [1,7,18]. In fact, Bennet and Karat [1] argue that finding ways to facilitate collaboration between
stakeholders to usability is a most urgent matter for HCI research.

Along these lines I suggest that researchers and practitioners should understand usability as a human activity rather than as a matter of methods and procedures. Such an understanding should help us focus on human activities such as collaboration and learning between stakeholders. It may also help us see stakeholders not only as professionals but as individuals who work together in professional or cross-professional groups.

The Participatory Design tradition, with its strong focus on collaboration, does perhaps best reflect the understanding of usability as a human activity [2]. Still, Participatory Design focuses on the beginning of the development process, where much has the form of sketches. It has not had much effect on the further development, such as for example the evaluation of prototypes of full-functioning systems.

I argue that understanding usability as a human activity will help researchers and practitioners bring the focus on for example collaboration and learning beyond the sketching phase of development, and into other parts of development relevant for usability, namely evaluation. In my opinion we need closer attention to stakeholders’ job roles and possibly conflicting goals so as to better understand how to support the successful interaction between stakeholders, such as developers, usability experts, and project managers. We also need to understand stakeholders not only as professionals but as individuals as well. Understanding usability as a human activity could help move the attention from the current somewhat inward focus to a broader and more outward one. As an example broadening the focus on the trouble of implementing usability work in organizations, or the methods used, to include factors such as how stakeholders learn about and understand usability issues, or how they collaborate to improve usability.

Such a broader focus will move usability studies away from lab-style studies that count how many problems a method identifies, or how many of the identified problems a developer plans to fix. Instead, an ethnographical approach is needed to bring qualitative and quantitative descriptions together to describe the forces at work when professionals conduct usability work in the industry. To give due credit, such work seems to be given increased attention, see for example [4,8,17]. Still, we need more field work that studies for example how usability work is organised and conducted in different and diverse organisational settings. We need to understand how ambitions and goals for the usability of products come to be, and how inter-personal and inter-professional relationships reflect on usability work in an organisation. Similarly, knowing more about precisely how great designs or design improvements came to be, would also be valuable to usability researchers and practitioners.

To answer all these questions (and many more) researchers need to move out of the offices, and into the field.

To conclude, usability is about UEMs, redesigns, problem severity and many other issues related to methods and results. Yet, I argue that usability work is mainly about human activities such as learning and collaboration. As a consequence improving usability is not mainly about getting more resources to test a system more often, or to study systems in more detail. Improving usability is about understanding usability work as a diverse collaborative learning process, where both professional and personal relations between stakeholders are crucial. To better understand the complex nature of usability work, and how we might improve downstream utility, researchers need to go where the action is - namely the industry - to do more studies.

REFERENCES


Two styles of presenting usability evaluation results

Timo Jokela
Oulu University & Joticon Oy
P.O. Box 3000
90014 Oulu, Finland
timo.jokela@oulu.fi

Niina Kantola
Oulu University
P.O. Box 3000
90014 Oulu, Finland
niina.kantola@oulu.fi

ABSTRACT
Two different styles of presenting usability results are described. The styles were used in a system development project where the evaluations were ordered and sponsored by the purchasing organization (city of Oulu), and the results were reported to the supplier companies.

Author Keywords
Usability evaluation

ACM Classification Keywords

INTRODUCTION
Communicating usability evaluation results effectively has been a focus of research during the last few years, e.g. [2], [7], [3], [4]. Common Industry Format, CIF, [1] has also been adopted to be used to report the results of a test of usability in summative format as defined in ISO 9241-11: effectiveness, efficiency and satisfaction in a specified context of use. However, Theofanos and Quensbery [8] remark that there is only little guidance in the literature for the good design of a report on a formative evaluation i.e. evaluations to improve a product during its design and development.

For better reports, general guidelines (e.g. [2]) have been given and sessions have been held (e.g. [6]) for exploring characteristics of useful and usable recommendations. Theofanos and Quensbery [8] also present a set of elements to be considered for a formative report. They have found that findings of usability evaluations can be organized in many ways such as by priority, topic, task/scenario, or screen/page. They may be illustrated by screen shots and other visual elements. Reports that are organized by page or screen often use screen images, either with callouts or to illustrate the explanation of the observations.

Jarrett [5] remarks that reports cannot do everything: “Just a good salesperson will do much of the selling before a formal proposal arrives, we should to think selling our findings to our audience” (p. 283).

We present two styles of presenting usability evaluation results. Our specific aim was to communicate the results as clearly and constructively as possible. Thereby we wished the evaluation results to lead improvements on design.

The specific setting of the evaluation was that the evaluations were ordered and sponsored by a purchasing organization (city of Oulu) and the results were reported to the suppliers of the system (system development companies). In other words, the suppliers of the system did not order the evaluations.

In this paper, we just describe our approaches for presenting the results. We do not yet have insight on their final effectiveness in terms of “downstream utility”.

SETTING OF THE EVALUATION
The city of Oulu started a system development project to enhance its health care services. For that, they initiated a joint project with system supplier companies. The system would be a health-care application that is used with a browser.

We were involved in the project when the first prototype of the system was available. The prototype consisted of only some of the (many future) services of the system. However, the prototype included the main structure and some sample services.

The purchaser of the evaluations was the city of Oulu. They were concerned about the usability of the system. They hoped that the suppliers would consider the usability evaluation results, and improve their designs.

The project consisted of two sets of evaluations (within one month’s interval). In the first one, our specific aim was to
communicate the results as clearly as possible. In the second one, the specific aim was to present results so that we would not insult the designers. The audiences (the suppliers) were different in these two cases.

THE STYLE OF PRESENTING THE RESULTS IN THE FIRST EVALUATION

In the presenting of evaluations, we had two starting points in mind:

• present the results visually
• also present the strengths

On the other hand, we did something that is not normally the practice in evaluations: we did not plan to propose redesign solutions. The reason for this was that we did not want to provide too much ‘free’ service to the suppliers. We wanted to make them clear that, in the long run, they need to pay for usability evaluations.

The results of the evaluations were a set of (PowerPoint) slides. The results presented as screen shots of the prototype where strengths were illustrated with green circles and an explanation in a box, Figure 1.

![Figure 1. Sample of presenting a strong in the design: green circle and explanation point (“breadcrumbs are used”) in the box](image)

The problems were presented with dotted line red circles and an explanation in a box, Figure 2.

![Figure 2. Sample of presenting a problem in the design: red dotted circle and explanations (“the meanings of the links do not distinct from each other”) in the boxes](image)

THE STYLE OF PRESENTING THE RESULTS IN THE SECOND EVALUATION

In the second set of presenting the evaluation results, we kept the visual screen shot capture style for presenting the results.

We, however, decided to not report the strengths and problems of the design, but just how successful the users were: whether they did perform the tasks correctly or not.

The reason for this was that we wanted to be careful not to insult the designers and the supplier company. As said, the companies did not ask for evaluation; we went to present the results to them without their order.

An example of a “user problem” is given in Figure 3. The part of the design is indicated with a circle (note – in this case the red color does not mean negative, it is neutral), with an explanation box.

![Figure 3. A sample of presenting a problem in user behavior (“none of the test users read this text”)](image)

An example of a successful user behavior is illustrated in Figure 4.

---

1 We used dotted line, in order to make the strengths and problems distinct also when printed black and white.
CONCLUSION
We showed two visual styles of presenting evaluation results

- One where strengths and problems of the designs were visually illustrated with explanations
- One where user behavior was illustrated (whether were successful or had problems when carrying out the tasks); but neither strengths nor problems of the design were explicitly identified.

The specific goal of the former one was clarity and communicativeness. The specific goal of the latter one was not to insult the supplier company.

It is still open how effective these approaches were in terms of downstream utility. The results were presented quite recently, and negotiations with the companies are still in process.

REFERENCES
Usability Case – integrating usability evaluations in design

Marja Liinasuo
VTT Technical Research Centre of Finland
Vuorimiehet 3, Espoo, P.O. Box 1000
FI-02044 VTT, Finland
marja.liinasuo@vtt.fi
+358 400 912711

Leena Norros
VTT Technical Research Centre of Finland
Vuorimiehet 3, Espoo, P.O. Box 1000
FI-02044 VTT, Finland
leena.norros@vtt.fi
+358 40 561 6774

ABSTRACT
In this paper we present a new method for usability evaluation. The method, Usability Case, defines usability of a particular system by a documented chain of inference including claims, arguments and evidence. With the help of this structure, repeated measurements can be accumulated and integrated. Usability Case is a tool for implementing usability in design, striving for downstream utility of usability evaluations. It is also a tool for monitoring human factors engineering in a complex design process.

Author Keywords
systems usability, complex systems, usability case

ACM Classification Keywords
Evaluation/methodology

INTRODUCTION
In this paper, downstream utility is understood as the practical effectiveness of usability evaluations on the designed system. Our usability evaluations focus on systems that are complex and safety critical. Hence, they have to be scrutinized from different points of view which we have tried to summarize in our concept of systems usability [3]. Furthermore, the design and implementation processes can be very long and are planned to have different phases. In one case, for instance, the design process continues even more than ten years. As a result, usability evaluations require a large number of evaluations.

Downstream utility of usability evaluations is a great challenge in this context. Designer’s problem is to structure and exploit usability information in his/her design process.

Additionally, managers need to be able to monitor the proceeding of the design process and to guide the necessary human factors engineering efforts, including usability evaluations. Furthermore, the designed systems are unique cases that cannot be compared with any other system, rendering the design process and project management even more demanding.

All these factors involve three partly interconnected methodological challenges. Firstly, the complexity of the system results in an abundance of information. To be able to comprehend the level of the usability of the system as well as to give appropriate weights for various usability elements calls for proficient information integration.

Secondly, as usability evaluations are performed during several phases of the design process, a lot of information of specific nature will be accumulated. Hence, this longitudinal information should be organized in such a way that information gathered in the past can efficiently be used in the design process throughout the design project.

Thirdly, as the evaluation focus is on unique singular design processes, case study methodology must be applied. In such a study, it is hard to find relevant references to usability evaluations. Consequently, internal criteria for good design, preferably based on theory, must be found to evaluate usability in an extensive and meaningful manner.

To meet these demands, we are currently developing a new type of approach to usability evaluation and design which is labelled as Usability Case. In this paper, a preliminary outline of this method is sketched.

THEORETICAL BACKGROUND
From Safety Case to Usability Case
A safety case is a requirement in many safety standards; explicit safety cases are required for military systems, the off shore industry, real transport and the nuclear industry. Furthermore, equivalent requirements can be found in other industry standards [1].

A safety case is "a documented body of evidence that provides a convincing and valid argument that a system is
adequately safe for a given application in a given environment” [1]. The idea of a safety case is to gather safety-related information into one document that is usable also later to demonstrate the safety of the system. Information is structured in an orderly manner that takes into account the abstraction level of the information as well as interconnections between the pieces of information.

Our proposal is that this way of organising information would be fruitful also in the context of usability evaluation. Given the specific demands that are set for usability by the design of a complex and unique system, the existence of a clear methodology taking into account the versatile nature of usability evaluations is severely needed. To obtain this, we used the basic structure of the safety case for building Usability Case.

Defining a Case
According to the definition by Bishop and Bloomfield [1], a case is a documented body of evidence that provides a convincing and valid argument that a system is adequate for a given application in a given environment. We widen this concept of a case by two perspectives. Firstly, while Bishop and Bloomfield focus on positive evidence only, we take into account also the negative evidence, i.e., also the problems of usability will be reported in the case description. It can be important to maintain this information so that it could be used as feedback to bring forth positive development in the matter in question. Secondly, Bishop and Bloomfield aim at producing a documented final description of the good performance of the system to be used during its operation. In addition to this, we see the need for usability information during the design process to guide and give feedback to design. Following these elaborations, we describe a usability case as creating an accumulated documented body of evidence throughout the design that provides a convincing and valid argument of the degree of usability of a system for a given application in a given environment.

As used in safety cases (Bishop and Bloomfield, 1998), three concepts are used to structure usability information in the Usability Case. Claims are entities that express the quality of the system in terms of usability. The abstraction level of a claim is high but various viewpoints and levels of detail are possible. Each claim comprehends of one or more arguments that are instrumental descriptions supporting the claim in question. Finally, each argument is based on evidence, that is, various data.

A case can be demonstrated in several ways. In the thesis of Kelly [2], the following options found in practice were as follows: describing the [safety] level through free text; tabular presentation structuring claims, arguments and evidence into separate columns in a table; claim structures that are built up from claims joined together in a hierarchical manner; traceability matrices that represent how one statement (claim, requirement, objective etc) is related to a series of other requirements; Bayesian belief networks that are graphical networks communicating the probabilistic causal relationship that exists between variables; and finally, goal structuring notation that graphically presents the structure of a safety argument by using goals of several hierarchical levels with constraints and solutions (roughly parallel with evidence).

The representation of claims, arguments and evidence seems to be as a clear and extensive way to describe the status of usability. Given the complexity of usability evaluations, free text seems to be a flexible means of describing the variety of usability attributes in a given system. Beyond this, also tabular or hierarchical structures are possible means to represent the usability case logics.

Characteristics of Usability Case
Usability Case (UC) describes how well the system in question fulfills the usability demands specific to that system. Understanding these demands both theory and practice are needed.

To obtain a systematic approach from the start, applying an adequate theory guarantees that UC is based on extensive and justified grounds. We have defined the concept of systems usability to indicate the appropriateness of a tool in aimed usage [3]. Our first premise is that systems usability must be defined context dependently. We maintain, furthermore, that systems usability is manifested in working practices, which are generic tool using patterns of the users established in the specific communities of practice [4].

To elaborate tool using practices we exploit the cultural-historical theory of activity, according to which a tool has three main functions. We used them to define the components of systems usability as follows: (i) instrumental function stands for the efficiency and effectiveness of the tool, which is the most used aspect in usability evaluations, (ii) psychological function denotes the cognitive structure of the use of the tool which is portrayed as fitness of the tool for human use; it also includes how satisfying it is to use the tool, and (iii) communicative function refers to the meaningfulness of the tool in social context. Additionally, we also consider user experience as an important element of systems usability. In particular, we are interested in how promising the tool appears in future usage. Of course, traditional usability concepts, e.g. by Nielsen, and corresponding breakdown of usability requirements are taken into account in our concept of systems usability.

Claims for Usability Case can be defined according to the analysis performed by the above-mentioned principles of system usability.

When Usability Case (UC) is produced throughout a long design process, the results of usability evaluations are accumulated in the UC. This means that in accordance with new phases of design, the claim structure will be completed.
and supposedly part of the claims shall be restructured to meet the new situation. This way UC is always up to date.

When UC is built as a longitudinal process, usability claims can be divided into at least two categories, that is, to those that represent good usability and those representing the poor one. In the design process, the amount of the poor claims may vary but in the end, their number and importance should be as low as possible.

Each claim is to be supported by at least one argument that describes on its part how the claim is demonstrated in practice. Each argument, in its turn, is justified by showing what data that argument is drawn from. In usability evaluations, that data can be ergonomic standards, results from questionnaires delivered to the employees, heuristic evaluations and so forth.

The basic structure of Usability Case is demonstrated in Figure 1 below. The concept and method of Usability Case is currently developed in three applications, nuclear power plant control room modernisation, in developing mobile information management for agricultural production and in civil command and control centre. In our oral presentation, we will elaborate the UC method in the last-mentioned context.

CONCLUSION
Theoretically, the method of Usability Case appears promising and fruitful for downstream utility of usability evaluations. By adopting the UC method in usability evaluations we take an active role in the design process and usability evaluations are integrated in the design. This is one way to improve downstream utility.

At the moment, we continue to develop the UC method both theoretically and by applying it in three complex domains.

REFERENCES

![Figure 1. Usability case, a demonstration of the structure.](image-url)
Facilitating usability work using different perspectives

Tobias Uldall-Espersen
Department of Computing,
University of Copenhagen
Universitetsparken 1, DK-2100 Copenhagen
tobiasue@diku.dk

ABSTRACT
Doing usability work is usually considered a multi-disciplinary task, but little is done to advance the multi-disciplinary side when obtaining and transforming usability insights into practical values in software development. This paper argues for a broader conceptualization of usability and suggests a number of usability perspectives. The aim is to support multi-disciplinary considerations among key stakeholders involved in software development projects where practical usability work is conducted. The usability perspectives evolved from an interview study involving 26 respondents in six Danish software development projects. Two examples of multi-perspective usability issues are given and it is described how usability issues relating to one or more perspectives under the right circumstances can crystallize in software development projects and have significant impact on end product usability. Finally, it is discussed how to apply the usability perspectives proactively when developing useful and usable software.

Author Keywords
Usability, Software Engineering

ACM Classification Keywords
H5 Information interfaces and presentation; H5.3 Group and Organization Interfaces; K6 Management Of Computing And Information Systems

INTRODUCTION
The aim of this paper is to develop an understanding of usability issues by use of different usability perspectives and to discuss possible, practical implications of such a view. The usability perspectives came into existence when reporting a study [11,12] based on interviews involving 26 respondents from six Danish software development projects. In [12] we argued that usability work in the case studies were oriented towards two different dimensions related to the various goals in the development project, among the stakeholders, and in the organization. The two dimensions found were: (1) usability work oriented towards the user interface or user interests and (2) usability work oriented towards the organization or other stakeholders. In [11] these findings were further analyzed and five usability perspectives were suggested. Based on the five perspectives, this paper analyzes one of the software development projects aiming at enlighten how multi-perspective considerations about usability issues increased the practical value of doing usability work.

The long-term goal of this research is to examine if different usability perspectives can be used to support and facilitate usability work, for example by evaluating software systems using different usability perspectives or by applying different usability perspectives when providing feedback from usability work. It is believed, that relying on different usability perspectives to understand usability issues could help grounding usability work in actual human values and such grounding could be a great advantage [1]. Such grounding could help building relationships with trust and confidence among key stakeholders participating in different way, which seem crucial to establish the proper condition for collaboration and therefore in the long term are important in order to impact software development projects [8,9].

METHOD
Data in this study originates from six case studies of independent Danish software development projects and involves 26 semi structured interviews [6] each of 60-90 minutes. Three cases were considered primary cases and the corresponding 14 interviews were transcribed and analyzed using open coding [10]. The remaining three cases were considered secondary cases and the corresponding 12 interviews were analyzed with the aim of supporting the findings from the primary cases.

The same interview guide was used at all interviews. It had four main themes: (1) The software development process. (2) Software quality. (3) Developing usable software. (4) General experiences with development of usable and useful software products. The interviews used a concrete software development project in the specific organization as a starting point and the respondents were not informed about the interviewer’s special interest in usability.

THE FIVE USABILITY PERSPECTIVES
The main observation was that the respondents discussed usability using different perspectives and five perspectives occurred repeatedly across the six software development projects. This section gives a brief introduction to the five perspectives. Further details are given in [11].

1. The interaction object usability perspective
Interaction object usability concerns whether users are able to successfully perform isolated interactions with user interface objects in the product.

2. **The task usability perspective**
   Task usability concerns whether the users are able to complete single tasks, i.e. fulfill a (sub) goal through a combination of interactions with user interface objects.

3. **The product usability perspective**
   Product usability concerns whether the product supports the users in reaching a coherent set of goals with effectiveness, efficiency, and satisfaction.

4. **The context of use usability perspective**
   Context of use usability concerns to what extent use of the system, possibly interplaying with other systems, in the actual context of use is effective, efficient, and satisfactory.

5. **The enterprise usability perspective**
   Enterprise usability concerns to what extent goals of the enterprise are fulfilled effectively, efficiently, and satisfactorily through use of the system.

The five perspectives are grounded in data from 26 interviews. It is not believed that the perspectives at this level of maturity constitute a complete and final framework of usability perspectives, i.e. that the definitions of the usability perspectives are fully adequate and no other usability perspectives could be found or added, but the five perspectives played a distinct role in the six studied software development projects.

It is furthermore worth noticing that usability issues can be related to one or more usability perspectives. When this happens suggested actions can be conflicting or complementing each other. When suggested actions are conflicting there is a risk of sub optimization, i.e. usability using one perspective is optimized at the cost of usability in other perspectives. Thereby the expected impact from addressing the usability issues might not be obtained and the expected value to the system not added. On the other hand, if one or more usability issues related to different usability perspectives are complementing each other, they might add a broader value to the system and in the end help producing a more solid, useful, and usable system.

**EXAMPLES**
Two examples from one of the primary case studies are given. The case study examines the development of a new internal insurance sales application called *Absalon*. Absalon replaced an old system called *MobilSalg*, which the users were very fond of, but it did not meet modern technical requirements and was replaced.

First example regards the use of a piggy bank icon. This example shows how a simple design issue evolved to a multi-perspective usability issue, where different usability perspectives suggested different actions. It could have added a serious usability problem to the end product, but the users involved in the development project realized the risk before it was too late and the problem was avoided.

Second example regards various data quality issues. This example shows how interplay between different usability issues related to different usability perspectives strengthened the development project and made the system more solid and usable. By continuously involving a multiplicity of key stakeholders, the usability issues crystallized in the project across different usability perspectives and the data quality issues became a core concern in the project.

**Example 1: The piggy bank icon**
*MobilSalg* used a lot graphical user interface (GUI) elements and a piggy bank icon was used to symbolize an ‘add discount’ option in the insurance information sheet. The top of Figure 1 illustrates how this could have looked like.

![Figure 1](image)

**Figure 1** – The figure illustrates three different ‘sheet of information’ models. In this example of a simple design issue, the use of a piggy bank icon and the ‘add discount’ option were related to three different usability perspectives.

The initial design suggestion followed an enterprise usability perspective, when the development team decided to remove the piggy bank icon and a number of other GUI elements. They wanted the system to appear professional and found a number of GUI elements childish and inappropriate. The interviewed user did not share this concern. He found the GUI in MobilSalg nice and useful when advising customers and considered the GUI in Absalon usable but improvable.

It was a concern in the development team that the system should be clear, intuitive, and easy to use and as a replacement for the piggy bank icon, the designer added an ‘Add discount’ button (Middle of Figure 1). Now this
Example 2: The data quality example
This example shows how a number of usability issues were related across different perspectives and involved different stakeholders. In this example the usability issues seemed to crystallize in the project and added a clear practical value to the end product. The following list describes a number of usability issues related to the data quality work. The list might indicate a possible ordering in time, which is unintended. It seems more likely that the understanding of the usability issues in the example arose and grew more randomly in the project over time and across key stakeholders.

- Following an enterprise usability perspective, data quality in the sales system needed to be improved. In MobilSalg about 8.000 errors were every year investigated and fixed by back office employees adding a significant cost to the sales process. This problem rarely informed the users since the errors occurred after the sales agreement was signed.

- Following a product usability perspective and a context of use usability perspective data quality also needed to be improved. Copies of customer data were held in locale databases and they were not always up to date. This could give faulty or insufficient pictures of the customers and could lead to embarrassing situations and missed sales opportunities.

- To improve product usability online wireless access to backend systems was provided. This removed the locale database problem, but introduced other problems regarding accessibility and performance. The online abilities furthermore enabled closer integration to other systems and other sources of data, and it provided the insurance agents with field access to other tools, such as email and calendar.

- As users now were writing data directly to the main insurance systems, data was validated instantaneously. This informed interaction object usability. Now the users needed to understand and accept rules for data validation of single input fields. The rules were not new, but they had not previously been enforced.

- This informed task usability since task completion time was increased and old habits became obsolete and needed to be replaced. Certain use practices that had developed over time were no longer allowed.

- One way of supporting the habit renewal was to reduce the flexibility of the system. This option was dismissed because it would reduce the context of use usability. The sales situations were very different and it was considered a priority to maintain flexibility with respect to different contexts of use.

- To increase task usability online access to the national civil registration number register was provided. This made it possible to enter a civil registration number and then retrieve the corresponding name and address information. This decreased the time spend on entering data during a sales.

- Since the users now spend less time on entering basic data, focus in the sales situation moved from the system to the customer and the context of use usability was improved.

- Enterprise usability was also improved as basic data was fetched automatically. The availability and consistency of data was increased and it became easier to establish household relations between customers. Combining customer data increased the possibilities for cross sale and the household relations were important for other systems in the organization, such as the data warehouse system and the customer relation management system.

DISCUSSION
The two examples shows how considering usability issues using different perspectives was natural in the software development process. Encouraging such considerations facilitated different stakeholders in contributing important insights and building up a shared understanding of possibilities and challenges in the project. This strengthened the collaboration between usability practitioners, developers, and other stakeholders and increased the practical value of the conducted usability work [4,9]. Furthermore, continuously involving domain/business experts and context of use experts, as well as technical experts and HCI experts, supported focusing on meaningful problems [3] and on maintaining relations to business goals, which previously has been reported lacking in professional usability work [2,7].

Having observed the five usability perspectives in a number of real-life software development projects raises the following questions. (1) Can the five usability perspectives be used to systematically facilitate usability work involving a broader set of stakeholders and (2) would it yield an advantage compared to other usability work approaches?

It is believed that the usability perspectives could be used to facilitate usability work, but this has not yet been tested and
it is not clear how to transfer the usability perspectives to applicable usability work techniques. The five usability perspectives might be naturally combined with many existing UEMs. Most UEMs address mainly the interaction object usability perspective, the task usability perspective and a few UEMs address the context of use usability perspective. The unaddressed perspectives, i.e. the product usability perspective and the enterprise usability perspective, could be seen as supporting perspectives, for example by use of heuristics or guiding questions, which depending on the situation could be more or less meaningful. From our observations, it seems that the usability perspectives most significantly stands out and adds practical value in cases with long-term involvement of various key stakeholders, easy access to key stakeholders, and efficient formal as well as informal routes of communication. This seems hard to reconcile with traditional usability evaluation methods and reported industrial practices [5]. There might be other ways to obtain and utilize insights from various key stakeholders not directly involved in the software development project [13], but whether a synergy effect as in the examples presented here can be obtained, without bringing people together, remains to be shown. Another possibility could be to apply the usability perspectives when working with feedback from usability evaluations in order to evolve a coherent set of redesigns appealing to a broader set of stakeholders. Workshops with various stakeholders were conducted in a number of the studied cases. Applying multiple usability perspectives to analyze feedback from usability evaluations in such multi-disciplinary workshops could be an option. However, if the usability evaluation feedback does not involve multiple usability perspectives, it might be difficult to obtain a valuable multi perspective outcome from the workshop.

A number of advantages could be expected when applying the perspectives and two possible advantages are argued here. First, the perspectives might facilitate the building of realistic expectations among various stakeholders. This is a serious problem, for example reported by Rajanen and Iivari [8] in a recent study where a significant usability investment became a total failure. The study by Rajanen and Iivari is a valuable case study because it shows how usability issues related to specific perspectives and specific stakeholders completely are ignored by other stakeholders using other perspectives. This problem was only realized when it was too late and usability then became a ‘curse word’ in the organization. Second, the perspectives can be utilized without establishing formal usability work procedures. Stakeholders in a number of the conducted case studies did not work systematically with usability, but still the usability perspectives were observed among the project team members. This suggests an opening for introducing more systematically work with usability in organizations not yet mature for larger usability investments.

ACKNOWLEDGEMENT
This work is part of the USE-project (Usability Evaluation & Software Design) founded by the Danish Research Agency through the NABIIT Programme Committee (Grant no. 2106-04-0022).

REFERENCES
ABSTRACT
In this position paper, I argue that a focus on problem counts in user-focused evaluation is a limiting view, particularly in relation to downstream utility. While fixing problems is an important consideration, a broader perspective that also seeks to develop systems that better fit their users – their ways of thinking and motivations, etc. – will result in new design opportunities that go beyond small evolutions of existing designs. I illustrate this view with a sketchy analysis of the use of a travel website.

Author Keywords
HCI evaluation; downstream utility; travel websites.

ACM Classification Keywords
H5.2. Information interfaces and presentation; D.2.2 Design tools and techniques

INTRODUCTION
In recent years, there has been an increasing focus on downstream utility as an attribute of any usability evaluation process [9, 16]. John and Marks [11] raised a related concern when they considered the ‘persuasiveness’ of the outputs from various evaluation methods. Underpinning these accounts, however, is an apparent assumption that evaluation is concerned with fixing ‘problems’ in a design and that persuasiveness depends on the quality of the evidence supporting the existence of the problem. As noted in the call for participation for this workshop, the results of user testing are often considered to be more persuasive (i.e. have higher validity) than those of any inspection method or analytical evaluation technique. Indeed, issues identified through inspection that are not also identified in user tests are often referred to as “false positives” (e.g. [12]). This leads to several challenges for those involved in evaluating interactive systems. In particular, user testing has many limitations:

1) Different evaluators note different difficulties from the same data, even think-aloud data [10]. Nørgaard and Hornbæk [14] provide one explanation for this as being that evaluators see what they are looking for and overlook that which they consider unimportant, or are not expecting. Lee et al. [12] highlight the importance of the evaluator’s prior expertise in identifying problems (in their case, when conducting Heuristic Evaluations). Within the context of usability practice, there are other influences on what is noted, such as the client’s brief and the broader context within which the evaluation is being conducted. User testing is not a reliable ‘gold standard’.

2) There can often be multiple interpretations of the underlying causes of the same surface behaviour, and so there is often a gulf between observed behaviour (‘problem’), explanations, and hence possible design solutions [15, 2].

3) When considering problem count, a variable considered important by, for example, Hartson et al. [9], there is a question about the level of abstraction of any problem description: a more abstract description will result in a lower problem count but might, conversely, cover many more actual and potential instances of user difficulties with a system. This point is illustrated by Cockton and Lavery [4] in their analysis of problem extraction.

4) People are flexible and adapt well to many system deficiencies. Green et al. [7, 8] developed a vocabulary for talking about many of these difficulties, called Cognitive Dimensions, which capture ideas such as a task that is conceptually simple taking many device actions (“viscosity”) or the order of actions that seems natural to the user being different to that demanded by the computer system (“premature commitment”). These kinds of difficulties can pass unnoticed, by both users and evaluators, unless they are primed to look for them in some way.

5) How people experience interactive systems depends on many factors including the context. This obviously includes the setting (laboratory or ‘real world’, with all its distractions and interruptions), but more subtly
includes how data is gathered and what participants are asked to do (e.g. task instructions).

In our discussions with developers (e.g. [3]), we have also found that the utility of a problem report depends on how easily the system can be adapted to overcome the identified difficulty. So it is not just what it reported, how it is reported, or the evidence that supports the report, but what the implications of that report are for those who are to use it.

All of these issues – and probably more – suggest that a reductionist approach to assessing usability and downstream utility is only ever going to reveal part of the story. While fixing problems (or avoiding them in the first place) is important, it may be limiting the vision in terms of identifying new design opportunities. I illustrate this point with a simple case study.

**Case study: evaluating a travel web site**

Four people who were making travel plans were asked to participate in a think-aloud study in which they were to find flights that would satisfy their up-coming travel needs. This was a compromise between a fully naturalistic study and a conventional laboratory-based study: the aim was to work with participants’ real-world needs (rather than some contrived task), but time constraints made it impossible to observe their actual travel-booking activities. The data was analysed using the first stages of CASSM [1], which yielded an overview of the concepts that people were working with. These included the following:

- That people were traveling between places, which would incidentally involve flying between airports: airports are less important than places. There are often alternative departure and arrival airports that serve travellers’ requirements. For example, there are four international airports in London and which one to fly from or to is often less important than other selection criteria. While some flight sites allow users to specify “LON” (i.e. any London airport), the same is not true for other areas, such as the region around Toulouse (with airports at Perpignan, Carcassone, Montpellier, etc.): there, the user has to specify precisely one airport rather than the region.

- People need to know not just about the airports, but also about connections from there to their actual starting points and destinations: this information informs their decisions about flights (e.g. what times of flights are practical).

- Costs – not just of flights, but of entire journeys – were an important consideration, as was total journey time.

- Individuals had many personal requirements, including: assurances about special meals; flights that do not transit via the USA (where transit is perceived by some as an unpleasant experience); a strong preference for direct flights; and flying with (or avoiding) particular carriers. Doubtless, a more in-depth study would have revealed a longer list of personal requirements.

There were also what would traditionally be considered usability “problems”, such as participants being unable to find the required options, having to select a return time for a one-way flight, accidentally closing the browser window and having to start again, and poor system response times.

Few usability evaluation methods would be likely to identify the mismatch between user requirements which centre around journeys between places and system representations which focus entirely on information about flights. Many flight purchasing sites do provide information about fares, flight times and transit points, but none (as far as we are aware) provide the more extended information and decision support that people wanted about journeys.

This is a simple example of a misfit between users and systems. It is not a “problem” that needs “fixing” because users have a rich repertoire of techniques – doing exhaustive searches, looking at paper maps, drawing on prior experience, etc. – that enable them to work with existing systems effectively. However, the interactive experience could be much better: there are design opportunities for integrated travel planning systems that support integrated travel planning, rather than just selling flights (with or without hotel bookings and car hire).

**CONCLUSION**

Many studies over the last few years have shown that there is no replicability in evaluation studies: the findings depend on the participants (for empirical studies); the evaluators [10] and their skills [6,12] and biases [14]; the method applied [2]; and subtle features of the systems being tested and the test setting. The receptiveness of developers to ‘problem reports’ or ‘redesign suggestions’ similarly depends on a wide range of factors including not just how these insights are communicated (with what evidence, by whom, in what form, etc.) but also what implications they have for design.

Downstream utility is, or should be, concerned with equipping developers to design better systems. A focus on reducing problem counts risks limiting the focus to ‘bug-fixing’. This appears to reflect a view of design as working towards the ‘perfect’ (bug-free) design solution where the role of evaluation is to make progress towards that solution.

An alternative view, articulated by Carroll and Rosson [4] in their work on the ‘task-artifact cycle’, is that every artifact creates possibilities, which define the tasks which are then easy or possible for users to perform, which in turn highlights new requirements for design. Before the advent of the Web and the development of e-commerce travel sites, the flight-finding activity described above would not have been possible without mediation by a travel expert: the evolution of designs to the current point have created new possibilities, but evaluations of those designs highlight shortcomings that, in turn, reveal new design possibilities. In this view, design and use co-evolve, and the role of evaluation is more than delivering a list of defects that need
to be fixed: it is providing new insights about people, systems, and contexts of working and playing, that suggest new design opportunities.

According to this view, downstream utility includes the provision of new design ideas as well as the identification of design problems. It is about possibilities as well as limitations.

ACKNOWLEDGEMENTS
The work reported here has been conducted as part of an EPSRC project GR/S67494.

REFERENCES
Maximising the uptake of usability results: a consultant’s view

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

INTRODUCTION
The workshop seeks to identify the optimal format and content for evaluation reports. There are no simple answers, as it will depend on the context of use of the report. There are many factors that will influence the extent to which the recommendations in a usability evaluation report are implemented. This paper is based on experience as a consultant. It briefly discusses how an evaluation report can be written to maximise the extent to which it meets particular stakeholder needs, and thus maximise the likelihood of implementation of the results.

BUSINESS CONTEXT
Quesenbery [7] has identified different categories of business or team relationship between the usability professionals and the people the report is delivered to, that should influence the content:

Introducing a team to usability creates a need to explain the approach, metrics and results carefully, as well as being careful about terminology

Establishing a new relationship suggests a need to establish credibility as you meet a team for the first time, as well as the possibility of needing to fit into an existing methodology or approach.

Working within an ongoing relationship implies that you are part of the team either on a full-time or consulting basis, and that you work on a series of usability tests or related work.

Reporting to an executive decision maker often means that the report gets special attention in some way. It also might mean reporting to someone who is not used to reading usability reports.

Coordinating with usability professionals is an emerging need, as there are more opportunities to share results across projects or over time; distributed teams have special coordination needs. It might also mean anticipating the needs of someone coming into the middle of a project and reading reports from previous usability work.

REPORT CONTENT
What content items should a report contain to maximise the likelihood that the problems identified will be fixed?

Method description
If the usability team is trusted, only a brief account of the method may be required, particularly if the report is to be produced quickly so that rapid decisions can be made. But if the report is to be circulated more widely, the method used should be described in sufficient detail to provide the readers with confidence in the results:

• For an expert evaluation, the description of the method should include information about the expertise of the evaluators, and an explanation of the approach taken to evaluation, with reference to any authoritative criteria used as a basis for evaluation (for example published guidelines or usability principles). (However experienced usability professionals tend to step though a task in the role of a user, relying on their previous experience to identify problems likely to be encountered by users. In this sense, they could be likened to a doctor diagnosing a patient.)

• For a user-based evaluation the description of the method should include an overview of the participant user characteristics, how they were selected against a pre-defined profile, how the tasks were chosen, whether users were given any assistance, and the extent to which the results from a small number of users are reliable.

Findings
All formative usability reports will contain a list of the observed problems, but these can be reported in a number of ways.

a) Positive as well as negative findings can be reported. The development team may be more receptive to bad news if this is presented in the context of the strengths of the product [6,7].

b) The findings can be prioritised by estimated impact on usability. Some sources recommend excluding some of the low priority findings, to provide the development team with a more manageable list [6]. But teams committed to usability may be interested in all the findings.

c) When possible, the priority for fixing problems should be estimated based on an assessment of the cost benefits of fixing each problem. This means that it may be uneconomic to fix some relatively high
impact problems, while some minor problems may be very easy to correct.

**Categorising recommendations**

There are examples of reports, particularly when based on heuristic evaluation principles, that categorise the results under each usability principle. In general, the development team will be more interested in the potential impact of the problem on usability, rather than the underlying HCI cause, so problems are most usefully categorised by page, screen or area in the application (except for problems that are common to a series of pages/screens).

However, if the development team is contributing to a heuristic evaluation, relating recommendations to HCI principles may help them learn to become better evaluators.

**Usability of the report**

Findings are most easily explained by the author and understood by the reader if they refer to an illustration of the problem. It is therefore highly desirable to include screen shots that illustrate the problems. This also suggests a report structure based primarily around individual pages/screens (or types of pages/screens).

**Statistical summaries**

Reports are easier to understand if results can be summarised in numbers, percentages or charts. However, spurious accuracy should be avoided, for example it would be misleading to say that 20% of users had a problem if this only represented 1 in 5. Making comparisons with small numbers can also be misleading, for example while a success rate of 80% might appear to be superior to 70%, statistical analysis might show that there is no significant difference.

Using statistical analysis with small samples is controversial, but is very helpful in giving an indication of the estimated confidence interval for results.

Including satisfaction questionnaires in user-based testing can provide very useful numerical results that can be compared across different products. This can provide more convincing quantitative support for the qualitative data obtained from think aloud and interviewing users.

Great care should be taken in interpreting any user performance data such as task time or success rate, if participants were prompted or in any way assisted by the moderator.

**Executive summary**

If the report will be read by a management audience, it is likely to have more impact if it has an executive summary that can be quickly scanned, which includes the purpose of the evaluation, overall strengths and weaknesses, and the key findings, for example summarising all the high priority recommendations.

**MAXIMISING THE UPTAKE OF USABILITY EVALUATION RESULTS**

Resources will be devoted to fixing usability problems by the development team if:

a) In the view of the responsible management, the benefits are perceived to significantly exceed the costs, or

b) In an organisation that manages risk during software development [4], not fixing the problems is perceived to result in a significant risk of failure to meet the project objectives, or

c) The organisation has a policy of using usability methods to minimise usability problems.

A rational decision would be based on a dispassionate assessment of the estimated cost-benefits of fixing particular usability problems, or on the estimated risk to project success resulting from not fixing usability problems.

The extent to which these factors influence the decision to devote resources to resolving usability problems will depend on the extent to which the decision maker understands the potential impact of usability problems. If the judgement of the individual or team producing the report is trusted, and there is a policy of minimising usability problems, a list of findings with estimated impact may be sufficient.

But it will generally be necessary to suggest a priority for fixing problems based on the estimated cost-benefit. Examples of potential benefits are given in Table 1 (adapted from [1]).

Table 1. Potential benefits of fixing usability problems

<table>
<thead>
<tr>
<th>A. E-commerce sales can be improved by increasing the number of web site customers who will:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Be able to find products that they want</td>
</tr>
<tr>
<td>2. Find supplementary information easily (e.g. delivery, return and warranty information)</td>
</tr>
<tr>
<td>3. Be satisfied with the web site and make repeat purchases</td>
</tr>
<tr>
<td>4. Trust the web site (with personal information and to operate correctly)</td>
</tr>
<tr>
<td>5. Not require any support, or use the web site for support rather than calling the support center</td>
</tr>
<tr>
<td>6. Recommend the site to others</td>
</tr>
<tr>
<td>7. Support and increase sales by other channels</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B. Product sales can be increased as a result of the usability of the product:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Improving the competitive edge by marketing the product or service as easy to use</td>
</tr>
<tr>
<td>2. Increasing the number of customers satisfied with the product who will make repeat purchases and recommend the product to others</td>
</tr>
<tr>
<td>3. Obtaining higher ratings for usability in product reviews</td>
</tr>
</tbody>
</table>
C. Employers can benefit from easier to use systems in the following ways:

| 1. Faster learning and better retention of information |
| 2. Reducing task time and increased productivity |
| 3. Reducing employee errors that have to be corrected later |
| 4. Reducing employee errors that impact on the quality of service |
| 5. Reducing staff turnover as a result of higher satisfaction and motivation |
| 6. Reducing time spent by other staff providing assistance when users encounter difficulties |

D. Suppliers and/or employers can benefit from reduced support and maintenance costs in the following ways:

| 1. Reducing support and help line costs |
| 2. Reducing costs of training |
| 3. Reducing maintenance costs |

There will be a particular need to identify the cost-benefits when significant effort would be required to fix a high impact problem. Depending on the development environment it may be more important to interpret the Table 1 issues in terms of risk to project success rather than just a business benefit.

**Obstacles to the uptake of usability evaluation reports**

The development team may resist the case for fixing usability problems for a number of reasons including:

a) The development team may question whether the reported usability problems will have an impact on actual users. The most persuasive evidence is for developers to observe a user-based evaluation, or to be shown selective video clips. If the usability results are based on expert evaluation anecdotal accounts of the difficulties that users have experienced with similar problems in other systems may be helpful. A management audience is often more influenced by examples than by a logical argument [5].

b) The impact may be accepted, but there may be reluctance to fix the problems due to the cost or consequent delay to the project release. In this situation the risks associated with releasing a product with poor usability need to be traded off against the risks of delaying the release, considering the impact on all relevant stakeholders.

c) The results may conflict with an established house style, brand, or the solution preferred by senior management. In this situation there may be no right or wrong answer: the implications for the key stakeholders of alternative solutions need to be analysed carefully to determine the best solution.

**REFERENCES**


Evaluating Procurement, Usability and Off-the-Shelf Office Software

Åsa Cajander  
Uppsala University  
Dept. of IT/HCI  
PO Box 337  
SE-75105 Uppsala, Sweden  
asa.cajander@it.uu.se

Elina Eriksson  
Uppsala University  
Dept. of IT/HCI  
PO Box 337  
SE-75105 Uppsala, Sweden  
elina.eriksson@it.uu.se

Jan Gulliksen  
Uppsala University  
Dept. of IT/HCI  
PO Box 337  
SE-75105 Uppsala, Sweden  
jan.gulliksen@it.uu.se

Abstract
Off-the-shelf office software has lately been receiving increasing attention in Sweden. This experience report illustrates different perspectives on a usability evaluation of the procurement process and the off-the-shelf software system. This study was part of a long-term action research project in a large government organization. Data was collected through documents, interviews, and participatory observations. This paper describes how the usability evaluation was carried out, how the context of the usability evaluation changed during these weeks, and finally discuss and interpret why the stakeholders experienced the evaluation as useful. The usability evaluation was found to be very useful by stakeholders from the organization, whereas the usability expert as well as our research group had difficulties in seeing the usefulness. Findings indicate an agenda that was revealed late in the project as one of the main reason for the importance of the evaluation. Also a true belief in the perfection of the off-the-shelf office software and a somewhat different view of what usability is might explain the downstream utility.

Author Keywords
Usability evaluation, use cases, software development, procurement, off-the-shelf software.

ACM Classification Keywords
H5.2. User Interfaces: Evaluation/methodology, User-centered design, Miscellaneous

Introduction
The purpose of this short experience report is to illustrate how a usability evaluation of an off-the-shelf office software could be perceived as useful or useless, depending on the stakeholder’s role. In the paper, we describe how the usability evaluation was carried out, how the context of the usability evaluation changed during these weeks, and finally discuss and interpret why the stakeholders experienced the evaluation as useful. This experience report illustrates a work in progress, that we find so interesting that we want to share the experiences so far.

Development of IT systems is an expensive business. Therefore organizations try as much as possible to cut costs and apply tools and techniques that are less expensive. As a consequence several systems suffer from low levels of usability and other quality deficits, both when it comes to the process according to which it has been developed and the resulting system. One natural consequence of such a cost awareness is an increasing interest for standard software systems.

The strive for rapid software development has created a great interest in the procurement of commercial off-the-shelf software (COTS) to manage developing time, cost and effort. However, often the integration work and the necessary modifications lessen the time and money savings and the procurement of COTS software might have other disadvantages as for example a dependence on the software manufacturer and usability problems partly due to a “too-rich functionality”[1]. Boem describes the pros and cons in this way: “most organisatopns have also found that COTS gains are accompanied by frustrating COTS pains” [1]. Hence, with an abundance of COTS software to choose from, the practical problem for many procurers is how to systematically evaluate, rank, and select software in the procurement process, and to address usability issues in this process. Some researchers have addressed this problem and suggest scoring techniques or metrics where usability is one quality feature included among many others in trying to compare different COTS software process, as for example [2, 3]. However, other researchers maintain that these scoring techniques are not valid as they are not appropriate for the analysis of complex software [4]. Researchers within HCI have proposed frameworks for matching user’s requirements to product properties, see for example [5, 6].
CASE SETTING
We did this study in cooperation with the Swedish public authority that handles financial aids for students. With offices in 13 cities in Sweden, the organization employs more than 1100 employees of whom 350 are situated at headquarters. Daily work at the authority is computer intensive, and the case handlers spend almost all day in front of computers. Computer support that is used in the authority is mainly developed in-house, and a majority of these IT projects starts because of changing legislations that influence the prevalent work practices. Moreover, a majority of the system used is developed in-house, and the IT department uses the largest part of the organization’s budget. Projects have participants from different parts of the organization, as in a matrix organization, where each department has their role and responsibility. Officially, all projects are run according to methods provided by the business development department, and the IT architecture department. These include methods for acquisition, project management, and systems development. In short, these methods provide a common framework, with descriptions of milestones, decision points, as well as templates and role descriptions.

About two years prior to this study, a large 3-year action research project (the AVI project) was launched within the organization with the purpose of increasing awareness and knowledge about usability and a healthy computerized work environment. The aim was to introduce and develop usability work practices based on our previous theoretical and empirical work [10, 11]. Focus is on increasing competence among all parties involved in developing computerized work through education and training, coaching and project cooperation.

Two usability experts work in different projects in the organization. These projects have both a procurer of the project, a project manager, and an IT architect manager that define the work to be done. Hence, the usability expert has three different managers, and these have different perspectives on the usability evaluation described in this paper.

THEORETICAL FRAMEWORK
Examining the role and interpretations of usability evaluation methods in system development practice requires an understanding of human action and what encompasses competence in human action. Our research has been inspired by the perspective that human action is situated [12]. Consequently, our research needs to take place in the actual practice. Our research is based on a constructivist and interpretive perspective, where we create and understand our reality by using language through communication. Interpretations are flexible, situated, and socially constructed. We adhere to the quality criteria and principles established by Klein and Myers [13]. Research based on case studies leads to contextual in-depth knowledge, and should not be generalized. We as researchers, the context, the organization, and the conditions, under which the research takes place, color the results. However, the organizations and the findings are not unique or unusual and therefore we hope that the reader will find the knowledge applicable in other settings.

METHOD
Our research is based on an action research perspective[14, 15], meaning that this research was carried out with the dual aim of action and research [16]. When working in action research projects we collaborate closely to people in the organizations, and this gives us the opportunity to see what is important to them and their perspective on IT systems development.

One researcher participated in the usability evaluation work, and had the role of a usability mentor to the usability expert. The mentoring role meant discussing and planning the usability evaluation as well as to guide the usability expert in her work. Moreover, documentations of the computer system as well as documentations of the requirements specification have been used in the usability evaluation. We also had short interviews with one the project manager of the AVI project.

During the interviews, a brief interview guide was used, and notes were taken on paper. After the interviews, data was analyzed using mind maps where main themes and ideas were highlighted. Finally, the interviews were discussed in relation to other findings from the participatory observations, and hence put in a wider organizational context. When writing up the study all names were altered, including the name of the software being evaluated, and quotations used are not transcribed verbatim.

RESULTS
The results section consists of a sequential description of the changing context and requirements of the usability evaluation.

The Introductory Setting of the Usability Evaluations
The organization has already an off-the-shelf office software for project management, here called Simplicity. Simplicity is a module based software, and the organization has decided to expand Simplicity with modules for activity management, monthly reports, and time reports. The main goals of the procurement of these new modules was to gain better insight in internal projects as well as achieving efficient planning, management and control of the IT-projects. The effects of the new modules in Simplicity are expected to be the following:

- “A potential improvement of the business development by 3-5 %

1 Case handler refers to a civil servant working at a public authority with case handling.
The procurement organization has worked with user-centered activities (as for example field studies and design workshops) in the procurement process of in-house system development. Hence, they do have some former experience of and competence in user-centered design methods. However, these have been used when procuring contract development, and never in the procurement of off-the-shelf software.

After having started to test the new modules of Simplicity in a pilot study, the manager of IT architecture and the procurer decided to make a usability evaluation of the system, as well as of the procurement documents preceding the pilot study. The downstream utility of such an evaluation would be the possibility to recycle the results from the evaluation when looking at other off-the-shelf systems as well as a better understanding of the use of usability evaluation methods when procuring off-the-shelf systems.

The procurer of the usability evaluation gave the usability expert a 15 minute description of the ideas behind the usability evaluation, described above. However, the objectives behind doing the evaluation was not at all clear to her and she found the setting somewhat confusing. At the time, the organization had no plans in neither improving the system nor cancelling or altering the procurement process. She described the situation in the following way in an email conversation:

“...I have started with the fuzzy assignment regarding the standard system Simplicity that we are going to use for time reports. Brian\(^2\) wants me to evaluate the procurement from a usability perspective. .../ But the task feels really odd since they have already procured Simplicity, and since it is already decided not to adapt anything in the system etc.”

Finally, the usability experts decided to do the usability evaluation in the following way:

1) A usability evaluation of the software Simplicity including
   - A heuristic evaluation inspired by Nielsen [17]
   - A think-aloud-evaluation with four users, and interviews with the same users. Two users were project managers using Simplicity in their project planning, and two were case handlers who were future users of the time report module.

2) A heuristic usability evaluation, and possibly an evaluation of future work environment problems, of the procurement documents including:

   - Requirements Specification
   - Use case describing time recording
   - Use case describing activity management
   - Use case describing monthly reports

The use cases of the procurement process that the usability evaluation addressed were a mixture of a requirements specification, and a use case. The documents were quite brief (4-6 pages) and contained a specification of what business process the new system must support. Despite the similarity with a requirements specification, the word used to refer to the document in the organization, was a use case.

The Changing Context of the Evaluation

The usability expert also came across different opinions of what usability is. These opinions might strongly affect the opinion of the downstream usability of the system. Some of the stakeholders at the IT department, as well as one of her managers (the IT architect manager), feel that a system is usable if it is possible to solve a task at all. In their opinion, Simplicity is very useful. Hence, they would expect the usability evaluation of the system to be quite positive. These thoughts were described in the following mail conversation where the opinions of the stakeholders is described by the usability expert:

“What is a usable system anyway? Take the time reporting in Simplicity as an example. People do their reports on a daily bases, and have learnt how to do it. The first page of the system is nothing they pay any attention to since they neither understand what the graphs show, nor what the words mean. However, they still manage to report their time, even though they do not understand more than half of the things on the screen. Is that a usable system? I know how to report my time since I have learnt how to do it in the system, but I do not understand what I do. “

However, after having done one think-aloud-evaluation with a project manager, the usability expert realizes that this usability evaluation will result in a very negative report:

“It is completely obvious, however, after the interview with Peter\(^3\) that there will be lots of problems when you release this system to everyone. It is difficult to find a system that is more complicated. /.../He had to go and look in the manual of the system to remember how to do different things. There were numerous complicated steps before he managed to solve the problem. Many tabs that he did not use or understand”

During this first week of the evaluation, the usability expert started looking at the documents from the procurement process. When doing this she found that the documents were not at all complete and included comments to the

\(^2\) Brian is an experienced systems owner that has written the procurement documents.

\(^3\) Peter is an experienced user of the system, and has worked with some modules of Simplicity for some years.
authors of the documents such as “Dan finds requirements on calculation regulations”. Furthermore, the version number of the documents indicates that they are simply a first draft, not final documents. This is interesting since they have already procured the system Simplicity, and the documents ought to be completed before the procurement. After having consulted different managers, the usability expert realizes that the procurement documents and the use cases were written after the actual procurement process. The reason for this is unclear, according to the managers and the usability expert. Due to the incompleteness of the procurement documents, it is quite impossible to perform a good heuristic evaluation of the documents, or an evaluation of the future work situation. The context of the evaluation has changed, and now the usability expert feels even more lost.

At this point, one of the researchers interviewed the AVI project manager about the usability evaluation of the system and the procurement process. His perspective was different from the usability expert’s, and the procurer’s perspective. In his opinion, the usability evaluation should focus on the case handler’s time recording in the system since this was the only part where any changes could be implemented. Here one should note that this was the only thing the usability expert had not begun doing so far. However, these suggested changes of the system could only be minor, as hiding functionality or graphs. There was no point in elaborating on major changes that would require much time. The major contribution of the usability evaluation, in his opinion, was recommendations to future procurement of standard systems. This is his perception of the downstream utility, and according to him the recommendations should include two main ideas: The necessity to look at usability when procuring off-the-shelf systems, and a written business recommendation stating that the supplier of the system should provide all changes and adaptations of the system.

DISCUSSION

In the discussion, we will elaborate more on the different stakeholders’ perspective on the usability evaluation, and interpret the context of their perspective in relation to downstream utility of usability evaluations. Both the usability expert and our research group thought that this evaluation seemed quite useless. However, other stakeholders to the evaluation valued it and the downstream utility.

One interpretation of why the stakeholders found this usability evaluation useful is that they were convinced of the excellent usability of the system. They wanted the usability evaluation to describe how brilliantly the system worked. This belief in the excellence of the system might have been related to the view of usability as describes in the quotation above about “What is a usable system anyway”. In this interpretation of usability, a system is usable if it is possible to complete a task even though it takes time, is frustrating and complicated. When having this view the usability evaluation would simply describe and point at the excellence of the system, since today many do use the system daily in other companies.

We believe that the usability evaluation is part of a agenda of one or several of the top managers, and they see the usability evaluation as something very useful downstream even though it might appear quite useless today. First, in the context of the strategic IT goals of the organization this usability evaluation has another meaning. The goals are called the Road Map and are written and envisioned by the IT architect manager. In the Road Map, all in-house-systems are replaced by off-the-shelf systems accessed through one common portal and user interface. The IT architect manager sees a great potential of the usability evaluation as it might contribute to a better understanding of procurement processes and usability when implementing the ideas of the Road Map. Secondly, he also adheres to the view of usability described in the previous section and consequently envisions a very good usability report of Simplicity. He might see this usability report as a good argument in the implementation process and discussion about the Road Map as it would stress the usability of off-the-shelf software. In his perspective, the downstream utility is obvious as it relates to his agenda.

The procurer of the usability evaluation has yet another agenda. This agenda is also connected to downstream utility as he thinks that it is possible to recycle the usability evaluation, and to use the material when analyzing another off-the-shelf system. He does not seem to realize that the usability evaluation is based on the work task of the user, and that these vary between different systems and products.

One perspective of the evaluation of the procurement process is that the stakeholders want to see how it is possible to work with usability in procurement of off-the-shelf systems. One should note that the usability evaluation of the process is done without any connections to the owners of the procurement process. Hence, at this stage they do not have any plans in integrating usability work in the procurement process of off-the-shelf software. One potential reason behind the vague assignment given to the usability expert regarding the process is that her time is free of charge within the organization. It is possible to assign tasks to her without considering return of investment, or concurrence with organizational goals. However, potentially there could be another agenda behind this assignment as well.

Here are some of the lessons we have learned from this case study:

- The usability expert did not feel confident about the heuristic evaluation as she experienced the outcome to be too dependent on her expertise. However, the stakeholders seemed to put great belief in the method as the objective truth.
• It is not easy to know which manager, or instruction to follow in a large organization with several managers at different levels. Hence, this experience report illustrates the importance of some written specification of what to do.

• The usability expert is taken hostage by the different managers in order to fulfill their agendas. She has no real power to impact the goals of the organization, and her expertise is used in a political game.

• At first glance, there seems to be not utility of this evaluation. However, in the eyes of the stakeholders there is an obvious downstream utility. Even though the usability evaluation might seem quite useless now, the experiences from it might contribute to better computer systems in the future.

ACKNOWLEDGEMENT
This work performed was sponsored by The Swedish Development Council for the Public Sector (Utvecklingsrådet). Moreover, we would like to thank CSN, and all the participants in the study.

REFERENCES
Upstream Futility Blocks Downstream Utility

Gilbert Cockton
School of Computing and Technology
University of Sunderland, St. Peter’s Way
Sunderland, SR6 0DD, UK
Gilbert.Cockton@sunderland.ac.uk

ABSTRACT
For much of the last century, relational ontologies guided progressive innovative thinking. This paper contrasts relational with essentialist approaches to evaluation and its role within an embracing development process. Downstream utility is argued to be largely dependent on upstream fecundity, which requires clarity of design purpose and constant co-ordination of all design, implementation and evaluation. Without both clarity and constant co-ordination, upstream futility can result.

Author Keywords
Downstream utility, evaluation methods, design purpose, worth-centred development (WCD) principles, ontology.

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
Downstream utility was proposed by John and Marks [10] as a key quality of an evaluation method, or at least, of one use of an evaluation method. There are two very different views of what is measured here. One sees it as measuring the quality of a method, the other as the quality of a method’s application. Let’s start as we need to continue. Such distinctions are fundamental to understanding what we are examining. Conceptual analysis is foundational to this position paper.

Wixon [13] implicitly proposes downstream utility as the main measure of evaluation. In contrast, Cockton [4] argues that downstream utility cannot be wholly due to use of a method. This debate cannot be resolved empirically. The fundamental issues are conceptual, and bring us up against a major rupture in the history of philosophy.

Relations spawn objects, beings and acts, not vice versa. [12, p.107]

Ontology is the branch of philosophy that studies being or existence, with a focus on what constitutes the basic categories and relationships of being or existence. In computing, the word has been trivialized to mean a single geek’s invented object structure (so is their pet collection a “zoology”?). Serres and Latour [12] assert that the fundamental category of existence is the relation, and not the object. They locate themselves within the twentieth century transition (heralded in part by Marx in the nineteen century) from essentialist to relational ontologies [11].

Essentialist ontology focuses on supposed immanent properties of objects. By being immanent (contained within, attributes of), such properties are atemporal, aspatial and universal. A rose is a rose is a rose, wherever it is and whatever you do with it. The question arises as to how we can know the properties of a rose, other than by observing its behaviours in specific contexts. Essentialist ontologies rapidly degenerate into a magical world of fetishes and totems [3]. We can only observe reactions, interactions and behaviours in context, and if these differ across contexts, then how can we locate within an object those properties that are atemporal, aspatial and universal? Are chairs strong (essentialist) or strong enough for their intended purpose (relational)? If they have any property at all, perhaps it is luck that only the right things sit or stand on them. A truck will generally be bad luck for any chair in its path (note generally, a relational, not essentialist, position).

To follow through with an HCI example, approaches such as guidelines and patterns tend to be essentialist, as they claim that interface properties follow from recommended features, irrespective of the usage context. While some uses of patterns may embrace context, they tend to do this so simply that they exclude dynamic relations between usage and the quality in use resulting from recommended features.

In a relational ontology, we recognize that what we may experience as properties, qualities or attributes of objects are in fact the result of relations with other objects (including a person’s subjective perceptions and world views). Hence for Serres and Latour, relations spawn objects, beings and acts. Thus colour is a relationship between an object, the light shining on it, and on some categorization system related to wavelengths of visible light. Similarly, weight is a relation between an object, the forces acting on it from much larger objects nearby (e.g., gravity of the earth) and a reference standard (e.g., weight of one litre of water). The case for a relational ontology is clear, but its intellectual consequences are very unpalatable for the rigidly scientifically minded, indeed for anyone who values an orderly, definable and predictable world. Relational ontologies steered twentieth century thought towards post-structuralism and postmodern
deconstruction [11]. While structuralist thinking asserted that relations between objects were the key to understanding existence, post-structuralist thinking asserted that there were no stable objects and hence no stable structures.

To provide an HCI example, the 1990s “turn to the social” in HCI demonstrated relational ontologies in use. The technological determinism of simple guidelines and patterns approaches was replaced by detailed evidence of how actual patterns, qualities and outcomes of interaction arise out of the relationships between interactive features, socially situated users, and the broader usage context.

Despite HCI’s continuing endorsement of its turn to the social, relational ontologies are not inherently superior. Essentialist ontologies run the risk of naïve grubbing for certainty, but relational ontologies can descend into nihilism and/or impotence. However, post-post-structuralist thinking is developing ways of overcoming such nihilism and absolute relativism. For HCI and usability, we need to be aware of these two distinct approaches to (proper) ontology. At the very least, we should know which class of ontology is implied by a position on downstream utility.

**ESSENTIALIST THINKING AND EVALUATION**

In HCI, conceptualization of downstream utility is essentialist whenever quality or qualities are seen as inherent in either an evaluation method or its use. The influence of a method and the outcomes for redesigned systems are seen here as almost immanent properties of the method and/or its application. The same holds for conceptualizations of evaluation method quality as ‘finding problems’, where again the yield and validity of a method’s evaluation reports are viewed as being due to immanent properties of the method itself. The technological determinism of many guidelines and patterns is replaced by the method determinism of the most simple minded assessments of evaluation methods.

An essentialist ontology can also be seen behind discussions of the ‘interplay’ between design and evaluation [5], where two separate and independent entities, ‘design’ and ‘evaluation’ have properties that determine the relationship between the two. Serres and Latour’s asserted relationship [12] is inverted: objects spawn relationships.

Now, there are clearly inherent limitations to some evaluation methods, which could be regarded as their ‘properties’, ‘defects’ or ‘attributes’. For example, Heuristic Evaluation (HE) tends to predict superficial problems that would require little, if any, interaction to emerge. HE misses most complex problems that only appear after perhaps minutes of interaction [7]. However, other apparent ‘qualities’ or ‘defects’ of a method have been shown to diminish or increase in relation to differences in method usage. Thus the use of extended problem report formats with HE was shown (surprisingly) to increase appropriate application of heuristics and to reduce false positives (largely by removing bogus problems with no evidence in the system) [8]. The general philosophical problem of how we can know absolute universal qualities of a specific entity without witnessing contingent interactions with other entities is relevant here: how can one establish the qualities of a method, except through using it in contexts that must influence its outcomes? Even changes to problem report formats have been shown to alter the apparent quality of a method (the real or true quality of a method being a fiction of essentialist thinking).

There is some value in essentialist thinking, since, for example, the persuasiveness of a problem report must depend on its content. Detailed and credible report content could be seen as a necessary condition for downstream utility, but not a sufficient one. However, this may not even be necessary. Some problems will have immediate face validity despite defects in reporting, especially when known to development team members and when due to knowingly substandard design decisions. Thus even when we find local qualities of evaluation methods in use, the impact of any qualities or defects will ultimately depend on the wider development context.

**RELATIONAL ONTOLOGIES AND EVALUATION**

Essentialist ontologies are a dead end, whereas relational ontologies could degenerate into a regress where everything depends on everything else. One way to avoid the ungroundability of post-structuralist thinking is to impose a closure. Such a normative step reduces everything from ‘anything possible’ to ‘only what is in scope’. Put simply, everything can’t be relative once we’ve taken a stance on some absolutes. We can do this because we are dealing with productive and not theoretical knowledge, a distinction originally made by Aristotle (whose third category was practical knowledge).

The purpose of a theoretical discipline is the pursuit of truth through contemplation; its telos [goal, end] is the attainment of knowledge for its own sake. The purpose of the productive sciences is to make something; their telos is the production of some artifact. The practical disciplines are those sciences which deal with ethical and political life; their telos is practical wisdom and knowledge. ([2, p.32])

We must thus take a position on software development. What is its purpose? What activities are involved? By fixing intent and composition, we can avoid potentially nihilist consequences of extreme relativism.

The purpose of software development is to implement a design of a system for a stated purpose and to test whether this purpose has been met. In so far as a system’s purpose is to achieve worthwhile practical ends, software development involves both productive and practical knowledge. Its purpose is clearly not theoretical knowledge: it may draw on theory, but it is judged by its practical outcomes. A normative subjective stance is thus legitimate, and thus the objective values of theoretical knowledge cannot dominate, whether descriptive or predictive.

In practical terms, this means that any discussion of downstream utility can only be framed with reference to a broader view of software development as a whole. Within this, we can look at the relations between five necessary components of software development, which we assert here to be: (a) the framing of purpose; (b) the conscious creation of an externally represented design; (c) the production of an
implementation; (d) the testing of this implementation relative to its purpose; and (e) corrective action required to bring future test results in line with the (revised) purpose for a system. The overall intent of such an inherently iterative development process is to produce a system that meets its intended purpose.

Usability evaluation is one aspect of testing of a system relative to purpose. It can have downstream utility in so far as it can identify failures of an implementation to meet purpose, and in so far as it can identify the causes of such failures. If it can do both, then it provides full support for subsequent iterative corrective action.

Now, for evaluation to identify failures of an implementation to meet purpose, the evaluation process must be focused on the achievement of such purpose. Also, for evaluation to profitably identify causes of such failures, an implementation must be related to a design that sought to achieve some purpose. If neither holds, only luck and/or chance will result in any downstream utility. High quality evaluation needs high quality designing, which in turn requires clear design purpose. A development process that lacks either clear design purpose (or clear design decisions focused on such purpose) will undermine downstream utility through its upstream futility.

**Upstream Futility**

Upstream futility requires defined as the absence of either or both of (a) clear purpose grounded in human worth, and (b) explicit design separate from implementation. If the antecedents of evaluation are fiddle, then evaluation is likely to be useless. In development contexts where purpose is vague and weakly coupled to worthwhile usage outcomes, and/or software is crafted incrementally rather than designed and then implemented, evaluation can have little value as a consequence [e.g., 1].

Upstream futility will block downstream utility regardless of the evaluation methods used and their ‘objective’ past performance. Evaluation can become a hopeless enterprise within a development process even before it has started. A range of factors can be seen to undermine the impact of evaluation [5]: a development process that makes rather than designs; a lack of resource for re-design, a lack of interest in re-design, and most generally, a lack of commitment to user-centred conceptions of product quality.

If we further extend upstream futility to include user-centred principles, we require further development activities in place before evaluation can fairly deliver. For example, we could require Gould and Lewis’ first key principle for designing for usability: *early focus on users and tasks* [9]. If we further require more demanding worth-centred principles [6], then the risk of upstream futility is considerable increased, since few development processes currently implement such principles.

**Upstream Fecundity**

Upstream fecundity requires the presence of both (a) clear purpose grounded in human worth, and (b) explicit design separate from implementation. In addition, upstream fecundity requires development to be *fruitful*, that is, (c) an installed system must deliver on its intended purpose (and ideally go beyond just delivering to ‘donating’ unexpected value). Upstream fecundity is thus the extent to which a development process can *bear fruit* by (over)achieving on design purpose through high quality human-centred design. As upstream fecundity is progressively reduced, upstream futility is eventually introduced in part, and ultimately, in whole. Evaluation methods cannot compensate for upstream futility; nor can they properly establish whether development has ‘borne fruit’ without extensive upstream fecundity. The first worth-centred principle of *fecundity* requires explicit framing of design purpose (this is not what is referred to as *requirements* in software engineering, but we expect purpose to evolve and tend to resent requirements changing).

Extensive upstream fecundity requires design purpose to be clearly *expressed* and well grounded through *sensitivities* to human worth. Designs need to be created on the basis of technical opportunities to *fairly and credibly* achieve the worth intended for each included stakeholder by a design’s purpose. Four worth-centred principles [6] of *sensitivity, expressiveness, bounded equity* and *credibility* must be well addressed to let evaluation deliver full downstream utility.

**Downstream Utility**

Downstream utility can be regarded as following from a final worth-centred development [6] principle of *improvability*, which subsumes Gould and Lewis’ second and third key principles for designing for usability: *empirical measurement* and *iterative design*. Improvability decomposes into subprinciples based on the RITE method’s three questions [13]: *evaluability (Is it a problem?)*, *understandability* (Do we understand it?) and *responsiveness* (Can we fix it?)

Evaluability requires that an implementation (or surrogates such as lo-fi or hi-fi prototypes, video scenarios etc.) can be assessed in terms of its destruction, degradation or delivery intended worth (or best of all, its donation of unexpected worth). This is stronger than Gould and Lewis’ [9] principle of *empirical measurement*, which fails to exclude irrelevant and inappropriate measurements.

*Understandability* and *responsiveness* are stronger than Gould and Lewis’ principle of [9] *iterative design*, which, unlike the principle of fecundity, does not require design changes to address design purpose. Changes could be made on the basis of irrelevant and inappropriate understandings of purpose. Also, these two subprinciples of improvability do not restrict iteration to design. Any component of development can be iterated, that is design purpose, the design or its grounding in sensitivities to uses and usage, and evaluation strategy and implementation.

**IMPLICATIONS FOR DEVELOPING**

Downstream utility can only be solely due to local evaluation qualities and defects if all other aspects of development are perfect. There are limits to what can be improved solely through evaluation methods and their use. The wider development context is critical to effective evaluation, which cannot fully compensate for defects elsewhere.

**Design Purpose and Evaluation Purpose**

The purpose of evaluation must be to assess the achievement of design purpose (anything else is a *usage study*). We need to
move far beyond the interplay between supposedly separate entities of design and evaluation to a unified view of development [6]. We cannot frame the purpose of evaluation in absolute essentialist terms. All qualities and defects are relative to design purpose. Usability should thus only be evaluated to the extent to which it is core to design purpose, just as design should be focused on the achievement of design purpose. Design and evaluation must be related to the whole of development, which in turn must create and maintain a focus on human worth.

The Need for Inspection as well as Testing

Development processes rather than designs should be inspected: testing cannot fully substitute for ongoing audit based on worth-centred principles. The quality of designing is relative to the achievement of worth-centred principles of sensitivity, expressiveness, bounded equity, credibility, improvability and fecundity. Empirical testing is required for the last two principles, with formative evaluation addressing improvability and summative evaluation addressing fecundity. The others cannot be ‘tested for’. Instead, associated acts and outcomes of designing need to be audited using human judgement, creativity and insight. A focus on the downstream utility of testing or usability inspection would be too narrow.

For sensitivity, inspection should assess how well design purpose is grounded in human worth (as needs, wants and dreams, and as real choices in real contexts) and also in technical possibilities and opportunities that could reveal latent wants or needs. The grounding of design purpose and design decisions (on technical materials, features and qualities) can be inspected to establish the quality of sensitivities. Without high standards here, evaluation may be condemned to assessing a design that may be usable, but not worthwhile.

For expressiveness, design purpose and design decisions need to be richly presented using a range of primary and secondary field evidence, as well as technical specifications that allow feasibility to be assessed. Here, the quality of both human and technical expression can be inspected. Without high standards of expression, evaluation may be condemned to assessing a design for worth that is poorly understood, or using technical features that are poorly implemented.

For fairness, achievable worth needs to be demonstrated for each included stakeholder (hence bounded equity). Fairness of likely outcome can be inspected. Without care here, evaluation may never assess a design from a proper range of viewpoints. For credibility, achievable worth needs to be established by inspecting the likelihood of worthwhile outcomes being achieved in use.

Designing (and development) should thus be sensitive, expressive, credible, fair, improvamoable and fruitful. Evaluation should be relative to normative principles, and not misguided by being conducted as objective tests of immanent properties of usability (or anything else) ‘within’ a system. Failure to plan for, and act on, these principles will contribute to upstream futility that must undermine the downstream utility of evaluation. In contrast, upstream fecundity achieved through worth-centred principles (or some preferred alternative) is necessary, but not sufficient, for downstream utility. Evaluation itself must deliver the remaining qualities sufficient for downstream utility, but these qualities are absolutely relative to, and dependent on, project specific elements of upstream fecundity.

CONCLUSION

Never examine the parts without knowing the whole and its basis. If design and evaluation are separated in a way that calls downstream utility into question, the answers do not lie in fixing design or evaluation in isolation, but in design management approaches to the whole development process. To work well together, design and evaluation must be planned and managed as parts of a coherent and well motivated whole. It is the overall development process that is effective: little can be effectively analysed or improved in isolation.

REFERENCES

ABSTRACT
In this paper we describe how ‘understanding and fixing’ usability problems must be preceded by ‘confidence and trust’ in usability problem reports, and provide sufficient detail to provide an understanding of the problem. One way of enhancing confidence and trust is through better understanding of how evaluators conduct evaluations and the role of comprehensive report formats that capture context and problem discovery/analysis justification. We present the DARE model of evaluator performance and explain how Extended Structured Problem Report Formats (ESPRFs) support the evaluator and encourage problem analysis leading to more reliable problem reporting.

Author Keywords
DARE model, ESPRFs, DCRs, Knowledge Resources

INTRODUCTION
Wixon [3] claims that the goal of usability evaluation is not just finding usability problems, but understanding and fixing them. This is a fair claim, but first of all we must have trust and confidence that any usability problem reported is a real usability problem. Wixon’s claim was a repost to research into the reliability of evaluation methods at the time. Such research relied on evaluating the reliability of evaluation methods on ‘how many’ usability problems they found regardless of whether the problems found were appropriate or valid. Indeed, commercially, the primary objective of evaluation is to understand and fix, or eliminate, usability problems. But when usability problems are predicted during inspection we must first consider their reliability. The following describes approaches that address this very important issue.

THE DARE MODEL

The DARE (Discovery Analysis Resources) model [1] explains the two stage process in usability evaluation. In its simplest form, evaluators first of all discover possible usability problems before confirming them as probable usability problems by analysis. During this two stage process the evaluator is reliant on a several distributed cognitive resources (DCRs) to help find usability problems and then analyse them (Figure 1).

Figure 1: The DARE Model

DISTRIBUTED COGNITIVE RESOURCES (DCRS)
These are the resources (knowledge, information or belief) that evaluators rely on to find and then analyse usability problems [6]. They are so called as they are not common in all evaluators. In fact, they may be absent in all evaluators until such times as they can access them. Appropriate cognitive resources support the evaluator in setting criteria and context for problem discovery and analysis. Below are examples of DCRs:

- Product knowledge
- Domain knowledge
- Design knowledge
- Technical knowledge
- User knowledge
- Task knowledge
Interaction knowledge

Evaluators need to have a variety of appropriate knowledge of the product and its goals, the domain in which it is intended as well as sound knowledge of user (abilities and experience), task and interaction knowledge. Such knowledge is essential to address criteria for decision making with regards to problem discovery and analysis.

Some evidence of DCRs comes from inconsistencies in evaluator performance. For example, not all evaluators find the same problems, not all evaluators miss the same problems, and evaluators do not all report the same false positives. Although there is a degree of overlap (some problems are reported by several evaluators), there are clear inconsistencies in both problem discovery and analysis.

If all evaluators had exactly the same cognitive resources, then surely there would be no differences in the problems reported by evaluators?

The truth is, inconsistencies and gaps in DCRs will reduce downstream utility through missed problems (problems not found at all, indicating inconsistencies/gaps in discovery resources) and false positives (a discovered problem but not correctly rejected through appropriate analysis, again indicating inconsistencies/gaps), and false negatives (a discovered problem incorrectly rejected indicating inconsistencies/gaps in analysis resources). We need to look at ways to reduce this, and one way to do this is via ESPRFs.

EXTENDED STRUCTURED PROBLEM REPORT FORMATS (ESPRFS)

This section describes the ESPRFs used in several studies by the author, and by several other researchers since their development. The report formats were originally developed as a research tool, however the use of the report formats can improve the validity of evaluator performance [2], and thus improve confidence in the problems reported. Furthermore, the detailed requirements of the ESPRFs also provide additional information beyond traditional reporting methods. This contributes to better ‘understanding’ of the problems reported, and supports understanding of the (in)appropriate use of DCRs.

Section 1 – Problem Description
Brief Description
Specific Likely/Actual difficulties
Specific Context (If Applicable)
Assumed Causes

Section 2 – Discovery Method
System Scanning
System Searching
Goal Playing
Procedure Following
Describe Method and Tasks steps involved

Section 3 – Heuristic Application
Heuristic Breached
Evidence of non-conformance

Section 4 – Confirmation/Exclusion Rationale
State clearly whether this problem has been kept or rejected
State specifically reasons for rejecting the potential usability problem

ESPRF – 1 Problem Description
This part of the tool was originally developed to aid the merging of problems when multiple evaluators are used to evaluate a system. There are four elements to this part of the ESPRF that provide multiple points of reference to facilitate accurate problem merging. A key issue with multiple evaluators problem reporting is the consistent coding and merging of problems. It is not unusual, in many cases, for evaluators to provide very different descriptions for the same problem. This makes accurate merging very difficult. With the ESPRFs the ‘master’ evaluator has 4 points of reference to make a judgment when merging problems. Moreover, providing details about the likely difficulties the problem will cause the user, the context in which it occurs and assumed causes of the problem contribute to a better understanding of the problem.

ESPRF – 2 Discovery Method
Describing the method involved in problem discovery reveals a considerable amount of detail as to whether an evaluator has adopted a ‘system centred’ or ‘user centred’ approach to the evaluation (Figure 2, below). The discovery method used also informs us of the typical DCRs that are likely to have been used in the strategy, and the appropriateness of DCR usage, should the problem be coded as a genuine problem or false positive.

Figure 2: DCRs and Problem Discovery

ESPRF – 3 Heuristic Application
This section is specific to evaluators performing heuristic evaluation as their primary evaluation method. However,
this section also provides valuable data on how heuristics aid evaluators (or not!) in problem discovery and analysis. Providing a rationale for heuristic breaches is another source for identifying DCRs.

**ESPRF – 4 Confirmation/Exclusion Rationale**

In an effort to gain maximum coverage of the DCRs used in evaluation, evaluators are required to give reasons for why a possible problem is kept as a probable one. Equally important, they are also required to give a rationale for discovered problems that they wish to reject as improbable ones. Further insight into the use of DCRs in analysis is achieved as evaluators justify analysis decisions, as well as allowing for the accurate coding of true and false negatives. A true negative is a discovered possible problem that is correctly rejected by appropriate analysis (inappropriate analysis would result in a false positive). A false negative is a discovered problem that is incorrectly rejected, but without ESPRFs, would be incorrectly coded as a missed problem.

It must be noted that such coding of hits, misses, true and false negatives can only be confirmed by falsification testing [5].

**CONCLUSION**

Wixon’s claims on the merits of downstream utility were, in part based on the then, current research into inspection methods that relied purely on problem counts. Simply counting problems found by an inspection does not provide an accurate assessment of a method. Simple problem counts do not take into account the actual frequency and severity of each prediction and missed problem [4], hence the limited confidence in potential usability problems predicted by evaluators.

Understanding and fixing usability problems must be considered separately. In order to fix them, we do need to understand them. But the question ‘can we fix them?’ is not the remit of the evaluation process. To ‘understand’ potential problems, we must first of all trust the predictions or findings of the evaluator. This can be achieved in part, by fully understanding the evaluation process (DARe) in the first instance, and the use of ESPRFs to both improve evaluator performance [2] and further investigate the influence of DCRs in evaluation.

Finally, it is fair to accept Wixon’s claim regarding the need to ‘understand’ usability problems because in a commercial environment the ultimate goal is to fix or eliminate such problems. However, with regards to evaluation itself, our first objective is to improve the quality of evaluation – not just count percentages! [4]

**REFERENCES**


How to Convince Stakeholders to take up Usability Evaluation Results?

Regina Bernhaupt
ICT&S Center, Univ. of Salzburg
Sigmund-Haffner-Gasse 18, 5020 Salzburg,
Austria
Regina.Bernhaupt@sbg.ac.at

Astrid Weiss
ICT&S Center, Univ. of Salzburg
Sigmund-Haffner-Gasse 18, 5020 Salzburg,
Austria
Astrid.Weiss@sbg.ac.at

ABSTRACT
In this paper we present our strategy to combine several usability inspection methods to increase the rate of usability problems solved in late development stages of industrial projects. By discussing the results in a workshop format with our industrial partners in several system development stages and confronting them with usability problems via live watching of a test situation we try to increase the awareness for usability matters. We call this methodological approach “evangelization-shocking” method.

Author Keywords
Usability Evaluation, User-Centered Design, Outcome, Results, Changes.

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
Academic institutions are conducting usability evaluations for several kinds of stakeholders, most often with industrial partners in quite late stages of the development cycle. Within the user-centered design process usability studies at late stages are producing high costs for changes [2], thus adoptions of results from usability evaluations are not likely at all in late stages of the development cycle. To escape the dilemma of only reporting a fast number of usability problems solved using various forms of usability evaluation methods and to increase the number of changes to enhance usability, we have started to develop a special industrial strategy. Following we will give a detailed description of how we started to use evaluation methods with industry partners and how we feedback usability problems to increase the number of problems solved within the next generation of the product. The case studies reported are rather informal than based on quantitative data but might give valuable insight in possible strategies to “evangelize” usability evaluation in industrial settings.

IMPROVING TAKE UP IN VARIOUS CASE STUDIES
Working in various projects on usability evaluation in areas like web, mobile and interactive TV, we have learned that depending on the methods used and combined within the project and the form of reporting results influences the number of usability problems solved. Based on these case studies we have developed an industrial method combination we found most useful during our practical work.

When asked to conduct a “quick” usability study for industry we are most often confronted with a very late development stage. The usability study is not seen as “increasing usability” but most often relates to “save the project”, “make the system work” or “prove that the system works” so it can be tested in the field. To clearly describe the goals that can be achieved in the usability test and to “evangelize” usability within the project we started to do a user and task analysis in all projects, even if we are just asked to evaluate the system in late stages.

During the user and task analysis we can investigate the initial goals of the industrial partner when developing the system, and the intended (marketing oriented) user groups. We typically discuss our findings in a workshop oriented setting, to find out peoples’ reactions. We started to describe the user groups using personas to help people understand whom they are designing for and to increase their involvement in the usability aspects during product development.

The user and task analysis is typically conducted within a week, depending on the material provided and the feedback times within the enterprises we work with. When discussing the results in the workshop we additionally refine the goals of the usability study, to make clear what kind of people we will be addressing in terms of usability, and how that might differ from the views of the design, programming or marketing team.
To prepare a better take up of the product we started to use a kind of trick. We ask the people working with us, to try to identify people they know, they work with or they even life with according to the personas described. People in the project team thus start to talk about people they knew, they start to argue that they think that their colleagues would for example immediately understand the user interface. If the level of involvment is high (meaning people really start to talk to their colleagues and friends how they would use the product), we even ask the participants to do so called short field trials. We especially use this form of short field trials in areas like web or mobile, were it is easy to access a web-based prototype from any place, or were it is easy to take the mobile phone with the prototype installation simply with you. Of course this kind of method is only useful when the security policy during product development is in line with our methodological set-up.

Using the more focused goals and based on the results of the user and task analysis we start a first heuristic evaluation. Typically the prototypes at that moment are not quite stable, interfaces are only partly implemented and material in general is not completely available. The heuristic evaluation results are presented in a workshop, showing on slides each of the discovered problem areas and showing in detail the interface or material used. We have found out that until this stage most of the people from the management department have not even seen the interface and what it is about. The workshop on the problem areas discovered during heuristic evaluation is typically the bottom in terms of negative emotional experiences for the participants from industry. A long list of possible usability problems is presented, most of the time even split up for the varying user groups (this is especially useful when working in the field of interactive TV, as the user groups vary extremely). During these, typically three hours long, workshops, we have heard most often that the system shall be thrown away, never will come to market, and related arguments.

Following Cockton [1] the goal of a usability evaluation would now be reached. The problems are found, described in detail and the design and development team could start to work on an iteration of the product. We can understand that from the perspective of a usability specialist, but in general we see our role more broadly. We started to integrate people from design and informatics, developing suggestions of how to improve the interface or product/system. We typically discuss the usability problems found within the heuristic evaluation, and then just make quite general suggestions what should be changed and looked at.

The next methodological step is a usability test -. We conduct a usability study on the (sometimes improved, sometimes not changed) prototype provided. The results of the usability test are again presented within a workshop. A video can be worth a thousand words. We typically categorize usability problems in more general clusters (e.g. usability problems related to navigation) and then show sequences of the test participants (typically failing).

We started to call the way we try to convince people to take into account the results of usability evaluations “evangelization-shocking” method. To increase the number of project members involved in the necessary changes after a usability study, we started to have members of the project team watching live the usability test. We made a download stream out of the current split-screen video produced during testing available via Internet. Additionally we set up a video conference to comment on what is going on during the test. We used this in a project for mobile interface testing, where programmers in France, China, Japan and the US were watching the usability tests. The timing for these tests was especially difficult, as we changed the testing times to early in the morning and late in the evening to enable participants from these country to follow at least 2 usability tests within their normal office hours.

Based on the results form the heuristic evaluation and the usability test we then start to suggest improvements within the prototype (and sometimes we even offer to support implementation and design). This involvement in the product development is far beyond what is typically seen as the role of a usability specialist, but in general increases the number of problems solved. When fully involved we typically manage to solve almost all problems, when only using the “evangelization – shock”- method we ended up in usability problems solved up to approx. 75 %. Of course the late stages of development we typically work in, can reduce the number of problems solved, but normally problems are not re-done in the next product generation. To support this “re-use” of usability problems found we started to inspect the interfaces and final products and compare what has been changed. We always include in the projects a so called “lessons learned” workshop, where we present the problems found and the problems solved within the final product. This is another step to increase the involvement of management, programmers and designers at the same time, after the stressful development process for the product is over.

SUMMARY AND OUTLOOK

Focusing on the take up of results from usability evaluation, we have started to develop a combination of usability evaluation methods, feedback workshops and other means like live-remote viewing of usability tests. Working in areas like web, mobile or even iTV we have learned that this combination of methods and approaches helps to “evangelize” usability in an enterprise. The method tends to evoke some bad experiences for the project members of the enterprise, but this “evangelization-shocking” method helps to support a critical reflection of ongoing processes and how to change them to reflect usability issues. Based on the workshop topic we now took up the idea to statistically analyze several of our projects (that were successfully
completed including the final “lessons learned” workshop), to see how many usability problems were solved within the prototype, within the first version of the product/system or afterwards.

ACKNOWLEDGMENTS
We thank all partners in our various projects for their cooperation and that they still work with us even if they now know that we simply try to evangelize them.

REFERENCES
Exchangeable Problem Sets Described with Evaluator Defined Constructs

Gudmundur Freyr Jonasson
University of Iceland
Hjarðarhaga 2-6, 107 Reykjavik, Iceland
gudmunj@hi.is

Ebba Thora Hvannberg
University of Iceland
Hjarðarhaga 2-6, 107 Reykjavik, Iceland
ebba@hi.is

ABSTRACT
In this paper we introduce a tool for recording usability problems in software into a database. The tool is designed to be flexible and the exchange of data between evaluators is simple with the use of XML (Extensible Markup Language) and well defined structure of the data. The tool enables evaluators to define constructs and store them in the database. The evaluator can search for existing constructs using a query language for XML called XQuery. The XQuery search function in the tool can also be used to search for usability problem descriptions. One of the main functionalities of the tool is the ability to merge duplicate problem descriptions. The tool compares construct values and makes suggestions whether some usability problem descriptions could be merged. The suggestions are rated by a relevance scale depending on how many constructs matched between different problem descriptions and therefore how strong the similarities are. We have performed analysis on three different problem sets to check for the comparison quality of the merging function in the tool. We are in the starting phase to find good ways to compare usability problems so the tool can be more accurate in grouping usability problem descriptions together.

Author Keywords
Usability, problem, construct definition, merging, exchange, relevance.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation] (e.g. HCI)
User Interfaces -- Evaluation/methodology.

INTRODUCTION
The project’s goals are to design and implement a tool which enables usability testers to record usability problems into a database using different usability evaluation methods. One of the tool’s requirements is to replace traditional methods of recording usability problems. Today most usability tests are done by recording problems on paper. In most cases there are several testers working together in groups so the paper work can be overwhelming after the user tests are finished. By using the tool the work after user tests should be a lot less and the exchange of data between evaluators is simple. The tool uses methods to diagnose what problem descriptions are unique and what descriptions are duplicates. The tool does this by comparing constructs within problem descriptions and if the comparison satisfies a predefined relevance scale the descriptions are grouped together. The user then makes the final call whether or not the two problem descriptions are duplicates. If the user decides that the descriptions are duplicates then those two problem descriptions can be merged into one description.

Figure 1 shows the use case diagram which presents a graphical overview of the functionality provided by the tool.

DESIGN AND DEVELOPMENT OF THE TOOL
The tool is designed to be flexible with the use of XML and all data storage is on XML format. Because of that, data exchange between different users is very easy, given the extremely good compatibility of XML. The tool is an online-tool, runs on internet browsers and connects to XML database called eXist [2] which is an open source database and is therefore free of charge. XQuery [3] is used for
searching the database which makes the search function very powerful and easy to use. Usability testers can also use the tool to define new constructs of usability evaluations by using the construct functions in the tool. There they can either create new constructs or search the existing ones using the XQuery search engine.

USE CASE DESCRIPTIONS
Construct Definition (Use case 5.1 and 6.1)
The user submits a new definition of a construct to the database (see figure 2). Each construct is described with coding dimension format presented by Gilbert Cockton [4] (Name of construct, Definition, Operationalization, Evidence Requirements, Sources of Bias, Motivation and Relevance, Used in and Examples).

The user can also search for previously stored construct definitions by using the XQuery search engine.

Construct Selection (Use case 2.1)
The user chooses the constructs he wants to include in his evaluation form (see figure 3). If constructs have already been chosen for a particular usability group by another user, then the user goes straight to describing problems and submitting. It is only necessary to choose constructs for a particular usability group once.

Submit Problem Descriptions (Use case 3.1)
After the user finishes typing a usability problem description he can submit the description to the database (see figure 4). The description is saved and a new empty problem description form is displayed with the problem number value increased by one.

Figure 2. Construct Definition

Figure 3. The Construct Selection form

Figure 4. Problem Description Form
Search for Problem Descriptions (Use case 4.1)
The user searches for problems with the use of XQueries (see figure 5). Given the power of the XQuery language, users can make very precise searches on the whole database, a particular usability group or single authors.

Find Similar Problem Descriptions (Use case 4.1.1)
The tool notifies the user if it thinks some usability problem descriptions are the same or alike (see figure 6). The tool does this by comparing constructs and if they satisfy a given relevance the tool groups those problem descriptions together and displays the results to the user. For example, relevance-scale A can be defined for CUP [5] and is about comparing the Trigger, Expected Phase and Failure Qualifier constructs. If for example four usability problem descriptions in a given set have exactly the same values in those three constructs, they are grouped together and the user makes the final call by evaluating if the descriptions are describing the same problem or not.

Problem Set Statistics (Use case 4.1.2)
The tool is capable of making simple statistical reports for problem sets (see figure 7). By using this function, the tool generates statistical information about the constructs and their values.

Figure 5. XQuery Problem Search. The user searches for the keyword “Umræða” (e. Discussion) in the A1 problem set.

Figure 6. Find Similar Problem Descriptions. Using Relevance A, the tool groups UP 16 and UP 56 together.

Figure 7. Example from the Problem Set Statistics Report
Save Results to XML File (Use case 4.1.3)
The user can export every XQuery result to a single XML file (see figure 8). This option makes it easy to work with the results elsewhere.

QUALITY ANALYSIS OF MERGING
We have performed analysis on three different problem sets to check for the quality of the merging function in the tool. We are in the starting phase to find good ways to compare usability problems so the tool can be more accurate in grouping usability problem descriptions.

Table 1 shows an example of a quality analysis. The problem set used is from user tests on the Owl learning management system [5]. The tool suggests seven groups of usability problems. We analyzed the quality of the tool by rating the tool’s suggestions either “Incompatible” or “Mergable” in column two. Of the problems listed in table 1, we found one problem in the Owl set that was not grouped correctly. The tool put problem number 59 in group nr. 5 (see table 1) which we rated as “Incompatible”. This particular problem should have been grouped (in group 3) with problems 16 and 56 because these three problems are mergable. It should be noted that the problems in the Owl set have been filtered and merged by a usability expert. The above exercise has been carried out to see if the tool can further identify more problems which can be merged. We have not yet analyzed the remainder of the problems in the Owl set to see if any problems were missed by the merging function in the tool.

CONCLUSION – FUTURE WORK
The design and development of the tool is still ongoing. To take better advantages of searching for problems, one possibility would be to code the descriptions according to keywords. Future development includes adapting the tool so it can be used by different research groups to perform advanced searches and to exchange data (cf. Code lights and Merging Study). Constructs of different research groups can relate to one another. As an example, a construct can be derived from another; a construct can be equal to another; a construct can be an aggregation of several other constructs, a construct can determine another etc. It would be beneficial to be able to express such relationships in the tool, so that we can write expressions about the constructs and their relationships [6].

Resource Description Framework (RDF) [7] ontologies can be written to represent relationship between constructs. For example, the prioritization construct can be a composition of impact, persistence and frequency.

REFERENCES
2. Open Source, Native XML Database. eXist: http://exist.sourceforge.net/


7. Resource Description Framework (RDF) http://www.w3.org/RDF/
ABSTRACT
This paper aims to further explore the notion of developer effect in terms of two aspects – persuasiveness and creativity. Three theoretical frameworks, viz. Information Integration Theory, Model of Arguments and Rational Choice Theory are reviewed; they consistently point to the fact that software development is a social process which is difficult to model, simulate and analyse. Recent research on open source software (OSS) has provided alternative views how software developers’ creativity can be enhanced. Similarly, modification of existing usability evaluation methods may also enable developers as well as usability evaluators to generate innovative redesign solutions.

Author Keywords
Developer effect, Information Integration Theory, model of argument, Rational Choice Theory, Open Source Software

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
Distinctions among programmers, software developers and software engineers seem nuance with the latter two normally being used synonymously. Whereas programmers invest most of their working time in program coding and debugging and occasionally “take over” design issues, software developers distribute their time and efforts among a cluster of tasks, ranging from writing specifications, quality assurance (QA) control, code reviews, to negotiating with executives and customers. In SMEs (small and medium enterprises) software developers play versatile roles; they are key actors deciding which bugs and usability problems (UPs) to fix and how; they are in charge of frontline programming tasks. In big corporate, project managers, who may not necessarily be knowledgeable in programming, are decision-makers in this regard. However, it is programmers who actually execute code rewriting, and the issue of filtering\(^1\) may lead to some unanticipated outcomes. In open source software (OSS) development teams, the dynamically evolving leadership democratizes this decision-making process, drawing in a variety of opinions; the efficiency may be lowered if the team is large and widely distributed, but may be compensated by a higher number of innovative enhancements. In summary, developers are professionals with a range of skills, knowledge and responsibilities being defined by the organizations and communities in which their work is embedded.

Presumably, the social dynamics in a SME, a large company or a community (an OSS group) influence substantially the effectiveness of software quality evaluation (i.e., downstream utility). This presumption is not new; since the dawn of the situated cognition movement in the late 1980s, the notion of “individual cognition” seems no longer legitimate but needs to be qualified by coupling it with contextual factors. Nonetheless, the core variability lies in certain personal attributes, which may be (dynamic ones) or may not be adapted (static ones) to situational demands (as shown in Figure 1).

---

\(^1\) Filtering refers to the effect of the journey a usability problem (UP) is placed from point of discovery to its destination, in terms of changes of recorded form, translations into formats, taxonomies, etc and passing between individuals involved in the development chain. The efficacy of usability evaluation method (UEM) of interest is affected by the number of times that information is interpreted, described and passed on, and the form and nature of the translation. This applies both to the way elicitation is set up to prompt observable phenomena, and the way in which descriptions and diagnostic judgments are formed, classified, shared and passed on. The UEM’s quality is affected by the reliability of inferences and the efficiency of these events (adapted from Mark Springett, 2007, accessible in MAUSE Wiki)
It is intriguing to understand which of these attributes are adapted and how the adaptation works. Specifically, when being provided with various sources of usability feedback, how a developer adapts her or his cognitive strategies to filter, prioritize, and judge the value of these data, which contextual factors she or he considers (or ignores), which earlier experiences and their associated emotional responses she or he recalls, and by what means she or he gains insights into redesign. Behavioral manifestations of these underpinning cognitive mechanisms are developers’ selecting UPs to fix with certain approaches. Understanding these mechanisms will enable the development of methods and tools to support effective and efficient incorporation of usability feedback into further improvement of the system of interest. Indeed, the primary aim of this paper to further explore the notion of “developer effect” (Law, 2006), which is defined as systemic biases of developers to select usability problems with particular characteristics to fix while leaving others unfixed. Developer effect is conceptualized as one of the many factors determining the effectiveness of usability evaluation. Note that developer effect per se is also a multifarious issue (Figure 2).

Specifically, I aim to investigate two major aspects of developer effect – persuasiveness and creativity. The former is inspired by the work of John and Marks (1997) and can be defined as the extent to which relevant stakeholders (incl. designers, developers, managers) are persuaded about the reality of the problems reported, and the necessity as well as the urgency to fix them; it is a composite construct being determined by several attributes, whereas the latter is inspired by the work of Hornbæk (in press) to see how usability evaluation can be exploited for idea generation rather than for defect identification. To achieve this aim, I review some relevant theoretical models (albeit not exhaustive) to derive some ideas and arguments. Note, however, that it is a work-in-progress; these thoughts entail refinements upon future inputs.

PERSUASIVENESS

Three theoretical frameworks are considered relevant to understand the issue of persuasiveness, viz. Norman Anderson’s (1996) Information Integration Theory (IIT), Stephen Toulmin’s (1958/2003) Model of Arguments (TMA), and Rational Choice Theory (RCT) (see e.g., Heath (1961), Elster (1986)). Whereas IIT is grounded in individual psychology, TMA and RCT are rooted in philosophy and economics, respectively.

Information Integration Theory

As mentioned earlier, the developer effect leads to a biased selection of a subset of UPS to be fixed. A source of bias is how developers judge the value of a usability evaluation report. That judgment is based on the perceived informativeness and credibility of the report content, and on developers’ trust in the expertise and experience of evaluators in case of predictive/analytical UEMs, and in the representativeness of end-users in case of empirical UEMs. This presumption is consistent with the established findings of classic persuasion research (Eagly & Chaiken, 1993) that persuasiveness generally increases with communicator (i.e., usability practitioner in our case) expertise.

John and Marks (1997) do not explicitly relate their notion of ‘persuasive power’ to any social cognitive theory. I believe that psychological theories of persuasion can shed light onto issues pertinent to the acceptance and realization of UEM outputs by developers and managers alike. When browsing the related literature, Norman Anderson’s Information Integration Theory (IIT) (1982) has caught my first attention. The cornerstones of IIT are four interlocking concepts: stimulus integration, stimulus valuation, cognitive algebra, and functional measurement. IIT aims to understand how multiple stimuli are integrated to produce a response, or given a response what effective stimuli are. Note that physical stimuli are rendered effective when they are transformed into their psychological counterparts through a chain of processing known as valuation operation. Further, humans frequently appear to be averaging or multiplying the stimulus information to generate a response. These psychological rules are generically termed cognitive algebra, which presupposes numerical representation of stimuli. While acknowledging the limitations of cognitive algebra in representing human thinking and action with absolute accuracy, I find it a practical means to approximate real psychological functioning.

In essence, IIT posits that different pieces of information in a multi-attribute judgment are integrated by an averaging process. Four general determinants of weight of information are: (i) relevance – defined as the implicational relation between the stimulus information and the dimension of judgment; (ii) salience – defined in terms of attentional factors; (iii) reliability – defined as the probability that the information is valid; and (iv) quantity – defined as the amount of information. The heavier the weight of the information is, the higher the likelihood it will be received,
accepted and yield the action (implicitly or explicitly) suggested. Empirical works by Anderson and his colleagues lend credibility to these weight parameters (e.g., Anderson, 1996). However, since its inception in the late 1970s, IIT has instigated a diversity of reactions, ranging from a very high appreciation to an outright reject. Nonetheless, I see the value of IIT in explaining the developer effect. One main theme of integration theory is the use of models as tools for cognitive analysis. Cognitive algebra can be a powerful tool for cognitive analysis because it dissects the observed response into its causal components. I applied the IIT to an empirical study on tracking down the downstream utility (Law, 2006). Figure 3 shows the mapping between the IIT determinants and usability evaluation attributes and their associated implications. Put briefly, these determinants cannot predict consistently developers’ fixing actions; developers tend to use such data as general references, and their final decisions on selecting UPs are shaped by other contextual factors.

![Figure 3: IIT determinants and usability evaluation attributes](image)

**Toulmin Model of Arguments (TMA)**

A significant implication of the TMA to the issue of downstream utility is formulating usability feedback as practical arguments. According to the TMA, people articulate practical arguments by first putting forward a claim of interest, and then providing justification for it, given that reasoning is a process of testing and sifting existing ideas. Toulmin asserts that a convincing argument can succeed in providing robust explanation for a claim, which will then withstand criticisms and reach an approving conclusion. Furthermore, Toulmin proposes six interrelated components for analyzing arguments, viz. claim, data, warrant, backing, rebuttal and qualifier. Table 1 illustrates a simple example how these elements are applied to analyse a UP, with the goal of persuading developers about its severity and the urgency to fix it. Presumably, usability feedback formulated in the form of a practical argument will be more persuasive than simply presenting a claim of a UP without any justification.

Besides, the format of data (e.g., written report, annotated screenshots, videoclips) play a role in determining the power of an argument. Some empirical work has been undertaken in this direction (personal communications with Mie Nørgaard in March 2007). Currently, I attempt to apply TMA to understand the consensus building process between developers and evaluators. It is well recognized that these professionals can judge UPs very differently (cf. evaluator effect; Hertzum & Jacobsen, 2001). How they draw consensus is an interesting and under-researched question to explore. Presumably, TMA serves as a good framework for analysing their discursive arguments.

**Table 1:** Applying Toulmin model of arguments to a UP

<table>
<thead>
<tr>
<th>Elements</th>
<th>Examples in Usability Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claim</td>
<td>The UP that users were not allowed to click the back button of the browser but only the given back button on the search engine user interface was severe.</td>
</tr>
<tr>
<td>Data</td>
<td>Users were annoyed that all data entered in the online form were lost when they clicked the back button.</td>
</tr>
<tr>
<td>Warrant</td>
<td>User frustration and significant loss of data and time are strong criteria for assessing a UP as severe.</td>
</tr>
<tr>
<td>Backing</td>
<td>According to the HCI literature (authority), these criteria are reliable and valid to attest the UP severity.</td>
</tr>
<tr>
<td>Rebuttal</td>
<td>Experienced users might be less likely to have this UP as they are more sensitive to constraints of a design.</td>
</tr>
<tr>
<td>Qualifier</td>
<td>Three evaluators consistently show a confidence level of 5 (highest) about realness and severity of this UP.</td>
</tr>
</tbody>
</table>

**Rational Choice Theory (RCT)**

Relevance of RCT to the study of downstream utility is its root in economics, which is related to the recurrent issue of return-on-investment (ROI) in the usability research and practice. From the perspective of economics, people are motivated by tangible incentives; formal methods have been constructed to model human profit-seeking behaviour. This has inspired sociologists and political scientists to build RCT, which attempts to explain all social phenomena in terms of “rational actions” made by self-interested individuals and sees social interaction as social exchange, modeled on economic action. The idea of rational action implies a conscious social actor engaging in deliberate calculative strategies, who calculates the likely rewards and costs of any action before deciding what to do (Scott, 2000). In other words, people exercise instrumental reason in making utilitarian choices. The advantages of RCT lie in its analytic strength, parsimony, and formalism (i.e., mathematical models), which, however, is also its pitfall.

Obviously, there exist various forms of value-oriented action alongside the purely rational types of action. In spite of criticisms leveled against RCT², the notion of payoff is deemed relevant to understand the issue of downstream utility. Given a set of actions and their corresponding payoffs, an individual is supposed to make use of this information to select an appropriate one. Otherwise, he or she is deemed irrational. Nevertheless, the assumption that we are able to know payoffs a priori or to apply the utility maximising principle (cf. bounded rationality) in cases of uncertainty is challenged. Other issues related to payoffs are: (i) Individuals having comparable backgrounds may

---

² For instance, it fails to explain people’s altruistic behaviour and trust as shown, say, in an OSS community, an example of so-called ‘gift economy’ or ‘gift culture’ (e.g., reciprocity/altruism)
not share a perception of a given choice set (cf. the assumption of description invariance; Tversky & Kahneman, 1992); framing of options (e.g., in terms of gains and losses) can systematically vary outcomes (e.g., preferences or actions); (ii) An individual may only be able get to know payoffs to actions through communicating with others; outcomes of these learning models, however, are rather unpredictable, depending on the effectiveness of such communication (i.e., providing/withholding/misinterpreting information about an unknown option or about payoffs of a known option).

Presumably, developers, who share similar background knowledge as well as experience and work under the same organizational and situational constraints, will demonstrate similar fixing strategies (i.e., selecting which UPs to fix) when being presented the same usability feedback. Payoffs in such a context are consequences of fixing or not fixing some UPs. The former case entails reasonable estimation of resources involved, which should be outweighed by the resulting improvements, whereas the latter case requires reasonable estimation of potential harm imposed by unfixed UPs. In both cases, selections of UPs can be informed by communications among developers and usability experts. Such a social transmission of utilitarian choices is an interesting topic to explore under the framework of RCT (personal communications with Edmund Chattoe-Brown in October 2007), especially when contrasting communication patterns being emerged in different types of social structure and social norms (i.e. a SME vs. a big corporate vs. an OSS). Interdisciplinary techniques such as ethnographic studies, computer simulations and social network analysis can be employed to model developers’ diverse perceptions of choice sets and decision mechanisms over time.

CREATIVITY

The notion that usability evaluation is not only for defect identification but also for idea-generation, providing insights for redesign solutions has become more recognized in the HCI community. Similarly, in the last five years, research studies on analyzing the open source software (OSS) movement have increased (e.g., Bitzer et al., 2007). It is reported that OSS development in some respects even demonstrates a higher speed of innovation than its commercial counterpart, providing competitive and viable alternatives.

One compelling question is what motivates highly qualified individuals to contribute to public goods at the expense of their private resources. The psychological theory of motivation by Deci and Ryan (1985, 2000) shed light on this issue. Accordingly, an actor is driven by intrinsic motives when she performs a deed that is inherently interesting, enjoyable or challenging for her. When she expects a separable outcome contingent on her deed, extrinsic motives come into play. The theory also asserts that external rewards may undermine an actor’s performance in a given task for which the actor has interest and, normally, talent. In the case of OSS, intrinsic motives include the OSS developers’ own need for software (i.e., user-programmer), fun/play, which is regarded as the most economically efficient mode of creative work (Raymond 2000). These mix with extrinsic motives such as job signaling and social capital (i.e. gaining recognition and social status in the community of interest/practice), and they altogether can be deemed as payoffs for OSS developers (cf. Rational Choice Theory discussed above).

The foregoing discussion points to the indispensable role of social interaction in software development, in contrast to the earlier studies focusing on individual developers’ characteristics (cf. Rasch & Tosi, 1992; Riemenschneider et al., 2002). As shown by an interesting study by Kidane and Gloor (2007), the communication structure among developers is an important factor influencing their performance and creativity. The authors find that adding more enhancements comes at the price of slowing down bug fixing and that a highly centralized communication pattern tends to lower creativity but increase productivity. Oscillation between hierarchical and decentralized communication structure is a strong indicator for creativity, but a more steady communication pattern favours productivity. Nonetheless, whether similar patterns can be observed in a commercial setting (SME or big corporate) is an interesting question to explore. These findings suggest that the (over)structuredness or over-management of the software development may kill the creativity of software developers. Can we argue that even standardization efforts and design patterns may have such undesirable effect by (over)emphasizing on rule or pattern following? Similarly, over-automation (cf. Norman, 1990, 2007; Linus Torvald’s critiques on debugging tools) may also take away creativity from developers.

Furthermore, not only developers but also usability evaluators should be supported to generate new ideas for redesign. Specifically, existing UEMs can be adapted with ideas derived from idea generation methods (IDM) to generate so-called hybrid approaches. For instance, claims analysis is to be augmented by DeBono PMI Method (1987); heuristic evaluation is to be integrated with Osborne’s (1979) checklist, and user tests are to be integrated with a modified 6-3-5 Methods (i.e., brainstorming approach) or DeBono’s Six Thinking Hat Method (2000) (quoted in Hornbaek, in press). It is intriguing to see whether the quality and quantity of ideas so generated will be higher than traditional approaches. We will conduct empirical studies to validate these ideas in the very near future.

---

3 Indeed, RCT is challenged by its failing to deal with the problems of collective action, social norms and social structure (Scott, 2000).

4 http://www.uwsg.iu.edu/hypermail/linux/kernel/0009.0/1148.html
CONCLUDING REMARKS
Recently there have been interesting discussions in the online forum PPIG (Psychology of Programming Interest Group) on agile methods. It is remarked in one post that a non-trivial amount of widely-used software applications have been developed through a process that is a random variation of canonical school-taught neat software engineering approach. It strikes me that software development is not at all a neat and clean endeavour as suggested by its components such as formal programming technique and systems development lifecycle. As remarked in another post in this PPIG thread “All methods are a set of heuristics to confront and manage the complex issue of software development. But what are the issues? … a person who knows the method, but does not have a solid grasp of the issues the method is to address, will have a failing project” (Clendon Gibson, 09/10/2007). One of the issues is definitely how developers communicate – a research question needs to be explored further.

REFERENCES
Case Study: Are CUP Attributes Useful to Developers?

Trausti Þorgeirsson
University of Iceland
Hjarðarhaga 2-6, 107 Reykjavik
traustt@hi.is

Marta Kristín Lárusdóttir
Reykjavik University
Ofanleiti 2, 103 Reykjavik
marta@ru.is

ABSTRACT
In this paper a case study of a classification scheme for usability problems called CUP (Classification of Usability Problems) [1,3] is described. The individual attributes are analyzed according to how helpful they are for developers to understand, prioritize and fix a defect. Additionally factors are analysed that determine whether developers decide to fix a defect or not.

The results show that developers found information on in what task the problem was found, the context in the UI (User Interface) where the problem was found and the description of the problem helpful in understanding, prioritizing and finding a solution to the problems. Other attributes like the severity rate was not rated helpful at all by one of the developers and only helpful for prioritizing the problem by the other developer.

The developers had fixed 7 out of 53 problems two months after the usability tests and had decided not to fix 33 out of the 53 problems at that time. The 13 remaining problems were on schedule still 11 months after the usability tests. Further research is needed to understand what can be done to get higher success rate for the usability problems being solved by the developers.

Author Keywords
Case study, CUP, usability problems, classification attributes, downstream utility

ACM Classification Keywords
INTRODUCTION
When conducting think-aloud sessions usability evaluators not often analyse the usability problems found in a structured way [2]. This could be because a systematic approach and a classification scheme is needed. This study was conducted to measure to what extent developers find useful the results of usability tests which have been analysed by a classification scheme called CUP (classification for usability problems).

The CUP scheme was developed by Hvannberg and Law [1,3]. The goal of CUP is to: “Classify Usability Problems (UPs) further in order to give the developers better feedback to improve their understanding of the UP, help them manage the maintenance of usability, enable them to find effective fixed for UPs as well as preventing such problems from reoccurring in the future”.

Three studies have been conducted using the CUP scheme. The first was in 2005 on The Owl (The University of Iceland’s teaching web), the second one was carried out in spring 2006 on Galvos at the Institute of Energy [3] and the current study. The first two studies were conducted to check the validity and reliability of CUP. In the current study the goal is to study what factors of the CUP scheme are important to developers when they decide which defects to remove and how to prioritize them.

MATERIALS AND METHOD
Ten usability tests were made, the results were analysed and described according to CUP and delivered to the developers. Two months later a meeting was conducted by the usability specialists to gather information on how useful the classification was to developers.

The usability test
Usability tests were conducted on a new version of software called Workhour. An old version had been in use for severral years, but in the new version the user interface was changed extensively. There are four main user groups of Workhour; ordinary users that work on shifts and those that work regular hours. The other two main user groups are managers that work on shifts and those that don’t.

The main tasks for ordinary users working on shifts is to check there monthly plan for shifts, ask for a day off and check if they have fullfilled all their work obligations for that month. The main tasks for regular users are asking for holidays and check if they have been too many hours off work. The Workhour system is very useful to managers, because they can do much of their organizing work in Workhour like check if all timestamps for their employees are correct, insert information about an employee that is sick and get an overview of how many have been sick over a particular period to name a few.
Ten usability tests were conducted by two usability specialists on the new version of Workhour running on a test database two months before it was installed. Five regular users took part in the test, four working on shifts in hospitals and one working on regular hours in a state institution. There were also 5 managers that took part, two working on shifts at hospitals and three working regular hours, two at a state institution and one at a software company.

Each user solved six or seven tasks in think aloud tests which were adjusted to their ordinary tasks. The total number of tasks in the study was 17. The tasks were made by one of the developers of the user interface that has good connections to the users. The user tests were conducted at their ordinary working place, so a lot of contextual information was also gained. Two usability specialists conducted the tests; one was the organiser and one the data recorder. Additionally everything that was said was recorded on tape.

**Analysing usability problems according to CUP**

From the verbal protocol and observation 53 specific usability problems were analysed according to CUP classification scheme in table 1. One usability specialist described the problems for the attributes: Defect ID, trigger, frequency, description, context and defect removal activity. Two usability specialists judged the impact factors individually; severity, failure qualifier and expected phase. On a meeting the mismatching rating was discussed until agreement.

Two versions were made of the usability problem lists, one long list with all the usability problems and another one where the problems had been split up according to the tasks it was found in. Additional information on the task was also added in that format. The main outcomes of the user tests were presented to the stakeholders at the software company; the project manager, developers and their manager. At that meeting the two developers involved in the study got the two lists each of the usability problems.

**Analysing what CUP attributes were useful**

Two months after the usability tests, the developers gave status on how many of the problems they had fixed, how many they planned to fix and how many they wanted to solve another way. For fourteen of the 53 problems developers filled out a form with 10 questions. On a five point scale, developers were asked to rate three things. First, how confident they were that the described problem was really a usability problem. Second, they were asked how well they understood the problem. Third, they were asked to prioritize fixing the problem. Finally, they were asked what CUP-variables were useful for understanding the problem prioritizing it and thinking up a solution for it. These results were gathered on a meeting with the developers.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defect ID</td>
<td>Identification number</td>
</tr>
<tr>
<td>Frequency</td>
<td>Number of users/experts that experience/product a UP</td>
</tr>
<tr>
<td>Trigger</td>
<td>Describes what an user is doing when she discovers the UP i.e. task scenario, heuristic, reflective question</td>
</tr>
<tr>
<td>Context</td>
<td>Describes in what part of the user interface the use/expert was when the UP occurred</td>
</tr>
<tr>
<td>Description</td>
<td>Concise description of the UP</td>
</tr>
<tr>
<td>Defect Removal Activity</td>
<td>Usability Evaluation Method e.g. user test and heuristic evaluation</td>
</tr>
<tr>
<td>Severity</td>
<td>Indicates what effects the UP had on the use/expert: Severe, Moderate, and Minor.</td>
</tr>
<tr>
<td>Task Efficiency (%/min)</td>
<td>Only for UPs found using user tests (UT), calculations based on the tasks scenario that the user was performing.</td>
</tr>
<tr>
<td>Task Effectiveness Mean time on task (min)</td>
<td></td>
</tr>
<tr>
<td>Task Impact</td>
<td>Mean time on task (min)</td>
</tr>
<tr>
<td>Instances of Prolonged</td>
<td></td>
</tr>
<tr>
<td>Instances of Help Sought</td>
<td></td>
</tr>
<tr>
<td>Failure Qualifier</td>
<td></td>
</tr>
<tr>
<td>Expected Phase</td>
<td></td>
</tr>
<tr>
<td>Actual Phase</td>
<td></td>
</tr>
<tr>
<td>Type of Fault Removed</td>
<td></td>
</tr>
<tr>
<td>Cause</td>
<td></td>
</tr>
<tr>
<td>Error Prevention Technique</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 List of CUP attributes and their values. The Post-CUP attributes are highlighted in grey.
RESULTS
In this section statistics are given on how the 53 usability problems gathered in this study were categorized. The CUP scheme has mainly three sections, descriptive attributes including description of the problem, how many users experienced the problem and in what context in the user interface the problem occurred. Secondly there are three subjective attributes that the evaluators judge, the severity rating, the failure qualifier and the expected phase. Third there are the Post-CUP attributes which are filled in by the developers when the problem has been fixed.

The Descriptive Attributes
In table 2 the frequency of users experiencing problems is shown. Most of the problems are only experienced by one user. While describing the usability problems the level of detail was kept very high, so problems were not joint if there was some slight difference in the way the users experienced the problems.

Table 2: The frequency of users experiencing a problem

<table>
<thead>
<tr>
<th>Problem experienced by</th>
<th>Number of problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 users</td>
<td>1 problem</td>
</tr>
<tr>
<td>4 users</td>
<td>1 problem</td>
</tr>
<tr>
<td>3 users</td>
<td>1 problem</td>
</tr>
<tr>
<td>2 users</td>
<td>5 problems</td>
</tr>
<tr>
<td>1 user</td>
<td>45 problems</td>
</tr>
<tr>
<td>Total</td>
<td>53 problems</td>
</tr>
</tbody>
</table>

It has been suggested in [5] to split the usability problems up in two categories: low-frequency and high-frequency. High-frequency defects are those which four or more users experience and low-frequency those below that. When using this definition we have 50 low-frequency and 2 high-frequency defects. Another idea is to classify high-frequency defects as those which more than one user experiences. That categorization gives 8 high-frequency defects in this study.

The trigger describes in what task or tasks the problem was found. From that information the frequency of problems found in each tasks can be calculated and is listed in table 3.

Table 3: List of number of problems found in each tasks

<table>
<thead>
<tr>
<th>Total number of problems per task</th>
<th>Frequency of tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 problems</td>
<td>1 task</td>
</tr>
<tr>
<td>8 problems</td>
<td>1 task</td>
</tr>
<tr>
<td>6 problems</td>
<td>2 tasks</td>
</tr>
<tr>
<td>5 problems</td>
<td>1 task</td>
</tr>
<tr>
<td>4 problems</td>
<td>5 tasks</td>
</tr>
<tr>
<td>3 problems</td>
<td>1 task</td>
</tr>
<tr>
<td>2 problems</td>
<td>2 tasks</td>
</tr>
<tr>
<td>1 problem</td>
<td>3 tasks</td>
</tr>
<tr>
<td>No problem</td>
<td>1 task</td>
</tr>
<tr>
<td>Total</td>
<td>17 tasks</td>
</tr>
</tbody>
</table>

Most of the problems were in menu context, which are the links on the left hand side of the screen used for navigating in the system. These were confusing to the users, they mixed up words that are used for different things so they chose first a wrong link often because the word used was not familiar to them.

The Subjective Attributes
In table 5 the results are shown for how the usability specialists rated the severity of the problems.

Table 4: Frequency of problems found in each context

<table>
<thead>
<tr>
<th>Context of the User Interface</th>
<th>Frequency of problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu</td>
<td>12 problems</td>
</tr>
<tr>
<td>Shift schedule</td>
<td>6 problems</td>
</tr>
<tr>
<td>Calendar</td>
<td>4 problems</td>
</tr>
<tr>
<td>Drop down lists</td>
<td>4 problems</td>
</tr>
<tr>
<td>Time records</td>
<td>4 problems</td>
</tr>
<tr>
<td>Links on the home page</td>
<td>3 problems</td>
</tr>
<tr>
<td>Contexts with 1 or 2 problems</td>
<td>20 problems</td>
</tr>
<tr>
<td>Total</td>
<td>53 problems</td>
</tr>
</tbody>
</table>

As can be seen in table 3 it was most common to find 5 problems in a task, but the variation of problems found in each task is quite high.

The data on the context describes where in the user interface the problem was found. In table 4 the frequency of problems found in each context is listed.

Table 5: The severity rating of the usability problems

<table>
<thead>
<tr>
<th>Severity rating</th>
<th>Number of problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severe</td>
<td>22 (42%)</td>
</tr>
<tr>
<td>Moderate</td>
<td>20 (38%)</td>
</tr>
<tr>
<td>Minor</td>
<td>6 (11%)</td>
</tr>
<tr>
<td>Irrelevant</td>
<td>5 (9%)</td>
</tr>
<tr>
<td>Total</td>
<td>53 (100%)</td>
</tr>
</tbody>
</table>

These results show that the usability specialists rated more than 40% of the problems being severe and almost 40% of them as being moderate. Both of the usability specialists attended all the usability tests, so they had observed all the problems.

In figure 1 the frequency of each category of the Failure Qualifier is given. An interesting result here is that incongruent mental model and overlooked make up for half the cases. Other categories are used less. This means that in most cases, the user doesn’t notice a part of the interface, or has the wrong idea about how it works. In this category there is a significant overlap, in particular between missing, incongruent mental model and overlooked. The suggested reason here for this overlapping is that the user may expect some entity to be present in the UI or misses some entity
that is present because his mental model of the UI does not 
match with it.

**Figure 1: Frequency of problems in each category of Failure 
qualifier for Worktime**

![Bar chart showing frequency of problems in each category of Failure qualifier for Worktime]

The frequency of usability problems in each category of the *Expected* phase is shown in figure 2. Three categories are never used; requirement analysis, quality attribute analysis, and design. The functional requirements category is only used once. This implies that this attribute can be simplified.

**Figure 2: Frequency of problem in each category of the 
Expected phase rating for Worktime**

![Bar chart showing frequency of problems in each category of the Expected phase rating for Worktime]

**DECISION ON SOLVING THE PROBLEMS**

A new version of the software tested was done two months after the usability tests and installed at the hospitals and state institutions. At that point information on how much of the usability problems were fixed was gathered. In table 5, information on how the developers decided to react to the problems is listed.

**Table 5: Reactions to the problems**

<table>
<thead>
<tr>
<th>Reaction</th>
<th>Frequency of problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fix right away</td>
<td>7 problems</td>
</tr>
<tr>
<td>On schedule for fixing</td>
<td>12 problems</td>
</tr>
<tr>
<td>Will not be fixed</td>
<td>28 problems</td>
</tr>
<tr>
<td>Add to the help of the system</td>
<td>5 problems</td>
</tr>
<tr>
<td>Checking it out</td>
<td>1 problem</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>53 problems</strong></td>
</tr>
</tbody>
</table>

These results were rather disappointing, especially that the developers decided only to fix 7 problems and that they had decided not to do anything with more than half of the problems. The severity rating of the problems fixed were: 3 severe problems, 3 moderate and one minor, so the fixing did not seem to be in correlation to the severity rating of the usability specialists.

In private conversations with one of the developers and one of their managers, 11 months after the usability tests were conducted no additional problems had been fixed. The reason the manager gave was that there was very much shortage of people in the group.

**ATTRIBUTES INFLUENCING ACTIVITIES**

To do an indepth analysis of what CUP attributes of the developers considered helpful when deciding on solving the problems, the developers were asked to fill out a questionnaire for 14 problems out of the 53 problems found.

In table 6 and 7 the rating of how helpful each attribute in the CUP scheme is according to the two developers are shown. The percentages show how large a proportion of the 14 defects that the two developers considered helpful. In most cases the two developers agree. As expected, the attribute frequency does not help with understanding a problem or thinking up a solution but it does help with prioritizing it. Both developers believe that trigger, context and description are helpful with all three categories. One of the two developers does not think severity helps with prioritizing a problem which is somewhat surprising. Neither developer considers expected phase or failure qualifier helpful with understanding a problem. –That is probably normal since they both wrote that they found the text description of problems sufficient. Neither developer finds failure qualifier helpful in finding a solution for a problem. This is disappointing since failure qualifier should be easy to understand and relate to for a developer,
especially in this context. Perhaps, the developers do not find it helpful to know the exact nature of a user interface defect but only what caused it so they can directly pursue that in the code. That way, there is perhaps too much emphasis on the symptoms, at the cost of the cause in the developers’ minds.

According to this data, trigger, context and description are mainly what the developers use for understanding, prioritizing and finding a solution to a problem.

Developers’ rating of problem descriptions regarding confidence, prioritization and understandability

Table 6: The ratings of developer 1.

<table>
<thead>
<tr>
<th>CUP attribute</th>
<th>Understanding defect</th>
<th>Prioritizing</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>0%</td>
<td>71%</td>
<td>0%</td>
</tr>
<tr>
<td>Trigger</td>
<td>100%</td>
<td>93%</td>
<td>71%</td>
</tr>
<tr>
<td>Context</td>
<td>100%</td>
<td>93%</td>
<td>71%</td>
</tr>
<tr>
<td>Description</td>
<td>100%</td>
<td>93%</td>
<td>71%</td>
</tr>
<tr>
<td>Severity</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Failure q.</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Exp. Phase</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Table 7: The ratings of developer 2.

<table>
<thead>
<tr>
<th>CUP attribute</th>
<th>Understanding defect</th>
<th>Prioritizing</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>0%</td>
<td>71%</td>
<td>0%</td>
</tr>
<tr>
<td>Trigger</td>
<td>100%</td>
<td>93%</td>
<td>71%</td>
</tr>
<tr>
<td>Context</td>
<td>100%</td>
<td>93%</td>
<td>71%</td>
</tr>
<tr>
<td>Description</td>
<td>100%</td>
<td>93%</td>
<td>71%</td>
</tr>
<tr>
<td>Severity</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Failure q.</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Exp. Phase</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

As previously stated, the developers rated the defects on a 5 point scale regarding their confidence that a defect was indeed a defect, its prioritization and its understandability. Both developers agreed that they understood the defects well. The prioritization was also similar between them. Both developers put most problems in categories 1 and 2 (where 5 is most significant and 1 least significant). This implies that they don’t find the defects very important to fix. The priorities were low in spite of the fact that about half the defects were classified as severe. Even so, one developer thinks severity is an important classifier in prioritizing defects.

CONCLUSION

We have described results from a study using the CUP scheme. The developers rated what CUP-attributes helped them with understanding defect, prioritizing it and finding a solution for it. We found that both raters believe trigger, context and description are helpful with all three categories. Neither rater thinks failure qualifier is helpful in finding a solution for a problem. This is disappointing since failure qualifier should be easy to understand and relate to for a developer.

The developers also rated the problems descriptions regarding confidence, prioritization and understandability. We found that developers did not put very high priorities on the problems, which is a bit surprising given how many severe defects there were. They generally said that they understood the problems well but did not express a high confidence in the problems indeed being problems. The developers had fixed 7 out of 53 problems two months after the usability tests and had decided not to fix 33 out of the 53 problems at that time. The 13 remaining problems were on schedule still 11 months after the usability tests. Further research is needed to understand what can be done to get higher success rate for the usability problems being solved by the developers.

REFERENCES

European Science Foundation provides and manages the scientific and technical secretariat for COST

COST is supported by the EU RTD Framework Programme