First publ. in: Usability : gaining a competitive edge / IFIP 17th World Computer Congress, Montréal, Québec, Canada, 2002, pp. 39-55

Use and Reuse of HCI Knowledge in the Software Development Lifecycle

Existing Approaches and what Developers Think

Eduard Metzker[1], Harald Reiterer[2]

[1]DaimlerChrysler Research and Technology Centre, Software Technology Lab, HCI Research Group, P.O.Box 2360, D-89011 Ulm, Germany, eduard.metzker@daimlerchrysler.com,
[2]University of Konstanz, Department of Computer and Information Science, P.O.Box D 73, D 78457 Konstanz, Germany, harald.reiterer@uni-konstanz.de

Abstract: In this paper we give an overview of existing approaches for capturing HCD(Human-Centred Design) process and design knowledge. We present an alternative approach that aims at fostering the integration of UE (Usability Engineering) activities and artifacts into existing software development processes. The approach is based on six claims that are derived from an analysis of existing UE process models and requirements of software developers. Our approach is embeddable in existing process improvement frameworks such as the UMM (Usability Maturity Model) and is supported by a web-based tool. An explorative study that we have conducted with software developers from various software development organizations confirms the potential of our approach. However the study indicates that our approach is stronger preferred by developers with more experience in user interface design.

Key words: usability maturity, process improvement, human-centred design methods, computer-aided usability engineering environment

1. INTRODUCTION

The relevance of usability as a quality factor is continually increasing for software engineering organizations: usability and user acceptance are about to become the ultimate measurement for the quality of today's telematics applications, e-commerce web sites, mobile services and tomorrows proactive assistance technology. Taking these circumstances into account

human-centered design (HCD) methods for developing interactive systems are changing from a last minute add-on to a crucial part of the software engineering lifecycle.

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

2. EXISTING HCD PROCESS MODELS: THEORY AND PRACTICE

A review on existing literature and case studies of some of the HCD approaches most applied revealed a number of organizational obstacles encountered in establishing HCD methods in mainstream software development processes [5]. We summarized these problems in claim 1-3.

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

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

Claim ***2*** *Most of the HCD approaches do not cover a strategy, how to perform HCD methods and process models depending on the usability maturity of the software development organization.*

Most approaches assume that the usability maturity of the development organization is high.

One typical assumption is that experienced human factors specialists are available throughout the development team and therefore HCD methods can be performed ad hoc. However, recent research shows that even highly interactive systems are often developed without the help of in-house human factors specialists or external usability consultants [6]. Therefore HCD methods often can not be utilized because the necessary knowledge is not available within the development teams [1, 2].

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

Claim 3 *Integrating UE Methods into mainstream software development process must be understood as an organizational learning task.*

Almost all approaches do not account for the fact that turning technology-centered development processes into human-centered development processes must be seen as a continuous process improvement task [7]. A strategy for supporting a long-lasting establishment of HCD knowledge, methods, and tools within development organizations is still missing. A model is needed that guides the introduction, establishment and continuous improvement of UE methods in mainstream software development processes.

To learn more about the way that HCD methods are actually used in software development organizations we conducted a study with software developers who are working on design of interactive systems in a variety of domains[1] [6]. The study revealed that the organizations examined are practicing highly diverse individual development processes, however none of the UE development models proposed by [1, 8-10] are exactly used. Furthermore, the persons who are entrusted with the ergonomic analysis and evaluation of interactive systems are primarily the developers of the products. External usability or human factors experts or a separate in-house usability department are seldom available. Furthermore, few of the participants were familiar with methods like *user profile analysis* or

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

cognitive walkthroughs which are regarded as fundamental from a usability engineer's point of view.

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

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

From the analysis of the interviews we derived high level requirements for a software tool to support the improvement of UE processes. These high level requirements are summarized in claims 4-6.

Claim 4 *Support flexible UE process models*

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

Claim 5 *Support evolutionary development and reuse of UE experience*

It was observed that the staff entrusted with interface design and evaluation often lacks a special background in UE methods. Yet, as the need for usability was recognized by the participating organizations, they tend to develop their own in-house usability guidelines and heuristics. Recent research [11-14] supports the observation that such usability best practices and heuristics are, in fact, compiled and used by software development organizations. Spencer [13], for example, presents a streamlined cognitive walkthrough method which has been developed to facilitate efficient performance of cognitive walkthroughs under the social constraints of a large software development organization. However, from experiences collected in the field of software engineering [15] it must be assumed that, in most cases, best practices like Spencer's are unfortunately not published in either development organizations or the scientific community. They are bound to the people of a certain project or, even worse, to one expert member of this group, making the available body of knowledge hard to access. Similar projects in other departments of the organization usually cannot profit from these experiences. In the worst case, the experiences may leave the organization with the expert when changing jobs. Therefore, the proposed tool should not only support existing human factors methods but

also allow the organizations to compile, develop and evolve their own approaches.

Claim 6 *Provide means to trace the application context of UE knowledge*

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

3. EX ISTING APPROACHES FOR CAPTURING HCD KNOWLEDGE

HCI has a long tradition developing *design guidelines*. Their purpose is to capture design knowledge into small rules, which can then be used when constructing or evaluating new user interfaces. Vanderdonckt (1999) defines a guideline by a design and/or evaluation principle to be observed in order to get and/or guarantee the usability of a UI for a given interactive task to be carried out by a given user population in a given context.

A detailed analysis of the validation, completeness and consistency of existing guideline collection has shown that there are a number of problems [16] [17], e.g. guidelines are often too simplistic or too abstract, they can be difficult to interpret and select, they can be conflicting and often have authority issues concerning their validity. One of the reasons for these problems is that most guidelines suggest a general absolute validity but in fact, their applicability depends on a specific context.

Based on the above mentioned problems with guidelines a different approach for capturing design knowledge has been developed, called *interaction patterns [17])*. A pattern is described in terms of a problem, context and solution. The solution of a pattern is supposed to be a proven solution to the stated problem. Patterns represent pieces of good design that are discovered empirically through a collective formulation and validation process, whereas guidelines usually are defined normatively by a closed group. Guidelines are mostly presented without explanations or rationale. Another differences is that patterns make both the context and problem explicit and the solution is provided along with a rationale. Compared to guidelines, pattern contain more complex design knowledge and often several guidelines are integrated in one pattern. Patterns focus on “do this” only and therefore are constructive. Guidelines are usually expressed in a

positive and negative form; do or don't do this. Therefore guidelines have their strength for evaluation purposes. They can easily be transformed in questions for evaluating a UI. A typical example of a guideline-based evaluation approach is the ISO-9241 evaluator [18].

Another interesting approach to capture HCI design knowledge are *claims [19]*. Claims are psychologically motivated design rationales that express the advantages and disadvantages of a design as a usability issue, thereby encouraging designers to reason about trade-offs rather than accepting a single guideline or principle. Claims provide situated advice because they come bundled with scenarios of use and artifacts that illustrate applications of the claim. The validity of claims has a strong grounding in theory, or on the evolution of an artifact which demonstrated its usability via evaluation. This is also a weakness of a claim, because it is very situated to a specific context provided by the artifact and usage scenario. This limits the scope of any one claim to similar artifacts.

All existing approaches for capturing HCI knowledge have in common that their use is largely focused on a small part of development process for interactive systems, namely interface design and evaluation. They are still not integrated in important parts of the software development lifecycle such as requirements analysis.

4. THE EVIDENCE-BASED USABILITY ENGINEERING APPROACH

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

We define evidence-based usability engineering as follows:

- *Evidence.:* Something, such as a fact, sign, or object that gives proof or reasons to believe or agree with something.
- *Usability engineering:*

(1) The application of systematic, disciplined, quantifiable methods to the development of interactive software systems to achieve a high quality in use;
(2) The study of approaches as in (1).

- *Evidence-based usability engineering:*An approach for establishing HCD methods in mainstream software development processes, that uses:

(1) qualitative and quantitative feedback on the efficiency of HCD methods collected in projects and integrates this data across project boundaries for controlling and improving the quality and productivity of HCD activities.

(2) concepts of organizational learning for integrating HCD knowledge, gathered as in (1), into software development processes.

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

After performing a method it should be evaluated if the method was useful for the development context or if it must be modified or discarded. The modification of the method should be recorded and stored for later reuse. Once a certain body of HCD knowledge is accumulated in that way, we have sound evidence for selecting an optimal set of HCD best practices for given development contexts. By continuously applying this procedure of conducting, evaluating and adapting HCD methods in a number of projects, an organization gradually adapts a set of HCD base practices to a wide variety of development contexts. This directly contributes to the general idea of software maturity models such as UMM [2]. According to these models organizations are highly ranked on a maturity scale, if they are capable to tailor a set of base practices according to a set of constraints such as available resources or project characteristics to solve a defined engineering problem.

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

- A process meta-model, which guides the selection of HCD methods for a given development context and their integration in an overall software development process in a flexible, decision-oriented manner. The model must as well provide a strategy for evaluating, refining and capturing experiences for new development contexts thus promoting continuous process improvement and organizational learning in HCD.
- A semi-formal notation for structuring knowledge relating to HCD activities allowing to save it in an experience base. The experience base allows to keep track of documented best practices and their application context even if the underlying context factors such as processes, technologies, domains and quality standards are still evolving, A model is needed to formally relate the best practices of the experience base to a development context.
- A CAUSE (Computer-Aided Usability Engineering Environment) for managing the experience base and allowing to predict optimal sets of HCD methods based on the available evidence and the experience of the development organization.

- An organizational concept for deploying, maintaining and evolving the experience base.

4.1 The Evidence-Based Meta-Model

Our evidence-based model comprises a set of organizational tasks to support the introduction, establishment and continuous improvement of HCD methods throughout the whole software engineering lifecycle. It helps to manage and tailor the HCD base practices defined in usability maturity models such as UMM [2] and the related methods practiced by the development organization according to specific constraints of the respective project and the experiences collected by the organization. The meta-model is based on our findings of experience-based improvement of user interface development processes [5]. We refined the model to the extent that we now use a formal model to relate a defined development context to a set of best practices. So the selection of an optimal set of HCD methods is guided by a model of the development context and the evaluation results of the HCD methods in real projects within the organization. In more detail the model consists of the following four logical steps:

- Step 1: Define, revise and extend the reference model

In the first step the reference model for the organization's HCD approach must be defined. If no reference model exists in the organization, one of the existing frameworks such as the process descriptions of UMM [2] could be used as a starting point for a reference model. After performing one or more projects, the reference model, should be revised based on the experience collected in the projects.

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

In this step base practices are selected from the reference model to be integrated in the overall software development process. However, further important factors have to be considered, e.g. the type of system to be developed and project constraints like budget and schedules. This information must be mapped into a context model and guides the selection of appropriate HCD methods for the selected base practices. The HCD methods which have been selected for the improvement of the development process have to be integrated in the model of the practiced software development lifecycle and the project plan and complement the overall engineering process used in the project.

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

Generally, at this step in the model resources have already been allocated for HCD activities, e.g., a usability engineer was nominated, who is responsible for coordinating and supporting the execution of the various HCD activities

of the new process. However, the efficiency and impact of the proposed HCD methods must be increased by providing the development team with best practices, tools and reusable deliverables of past projects (e.g. templates for usability test questionnaires, results of conceptual task analysis or user interface mockups) which facilitate effective performance of the selected HCD methods.

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

The HCD methods applied in a project should be rated by the project team members. These ratings form the rationale for following projects to adopt or reject HCD methods. Methods poorly rated by project team members in a project have a low likelihood of being reused in later projects.
During the execution of HCD activities, artifacts with a high value for reuse are generated by the participants of HCD activities, for example, templates for usability tests, reusable code fragments, or an experience on how to most efficiently conduct a user profile analysis for assistance systems. Observations like these comprise HCD experience that have to be captured and organized in best practices along with the development context in which they apply to allow for easy reuse in the same or subsequent projects.

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

The outer cycle guides the improvement process on the organizational level. The organization's HCD experience base, available methods and reusable artifact are tailored to the needs of the projects. Experiences collected on the project level are used to improve and evolve the organization's reference model. Ratings of methods provided by project team members guide and simplify the selection of methods in subsequent projects.

4.2 USEPACKs and the HCD Experience Base

The main focus of the evidence-based Usability Engineering methodology presented in this paper is to capture and evolve HCD knowledge for reuse and process improvement. Therefore we have developed a semi-formal notation for structuring *process* knowledge relating to HCD methods. For this purpose we use the concepts of USEPACKs (Usability Engineering Experience Package) and a context model. While USEPACKs are used to capture HCD methods, the context model is used to

formally relate these methods to a development context. The main difference between USEPACKs, guidelines, interaction patterns and claims is that USEPACKs are primarily focused on *process* knowledge. The aim is to present context specific process knowledge, structuring the whole HCD process in different generic base practices. In these different base practices guidelines, interaction patterns or claims could be very helpful to represent specific HCI *design* knowledge. The design knowledge is embedded in HCD activities with the help of a set of reusable artifacts. Therefore a USEPACK encapsulates best practices on how to most effectively perform certain HCD activities and includes the related artifacts like documents, code fragments or templates that facilitate the compliance with the best practice described.

4.3 A Computer-Aided Usability Engineering Environment

To increase the impact of our approach we developed a CAUSE called ProUSE (Process centred Usability Engineering Environment). ProUSE consists of an HCD experience base and three logical components for planning, selecting, applying, evolving and assessing HCD methods and reusable artifacts.

The experience base is seeded with an initial set of best practices in form of USEPACKs. In our case we have adopted a variety of usability engineering methods from Nielsen and Mayhew [1, 8] but in general any HCD approach should be appropriate. The REUSE (Repository for Usability Engineering Experience) [6] component is used to capture, manage and evolve best practices related to HCD activities. It assists in documenting best practices using the USEPACK concept and relating them to a formal development context using a context model and storing them in the experience base. SATUP (Setup Assistant for Usability Engineering Processes) is used to plan HCD activities for a software development project. Given all available context information on the process (e.g. which HCD base practices should be performed), project (e.g. duration, budget, team size), domain and technology context factors, SATUP will propose optimal HCD methods and reusable artifacts based on the accumulated experience of the development organization stored in the experience base. If there are alternative USEPACKs available for a defined context situation, SATUP will compare the available ratings for all USEPACKs using a fuzzy multi-criteria decision making approach and propose the optimal USEPACK. Once an optimal HCD process was planned, CUES (Cooperative Usability Engineering Workspace) can be used by a distributed development team to access, perform and assess the HCD methods selected. The ProUSE prototype is based on Java technologies and integrated via a

web portal concept which makes the modules available through intranet and web browser.

5. DEPLOYMENT OF PROUSE IN AN INDUSTRIAL SETTING

Before the ProUSE environment is deployed in a development organization for the first time, the systems experience base should be seeded as described in section 4.3. After this first step, the experience base contains a set of USEPACKs that describe methods for each base practice of the reference model used. These USEPACKs will usually have a poor context situation. Imagine now that a large software development organization is developing assistive, interactive home entertainment components and has recognized that usability aspects will play a major role in a new project. In that project an intelligent avatar for assisting users in programming their video recorders should be created. The organization decides that the overall development process should be supplemented with usability engineering activities. One usability engineer is available for the project but the rest of the development team is inexperienced in usability engineering.

When the project is planned, the usability engineer uses SATUP to create a new project and maps the projects usability constraints to the context factors of the context model thereby creating a context situation for the project. After the usability engineer modeled the development context, the project is accessible to the development team.

Members of the development team can now access the new project via CUES. CUES creates a workspace where the USEPACKs relevant for the defined context situation can be accessed by the development team to support their usability engineering activities. CUES provides the development team with a tailored view on the experience base, shielding them from irrelevant information.

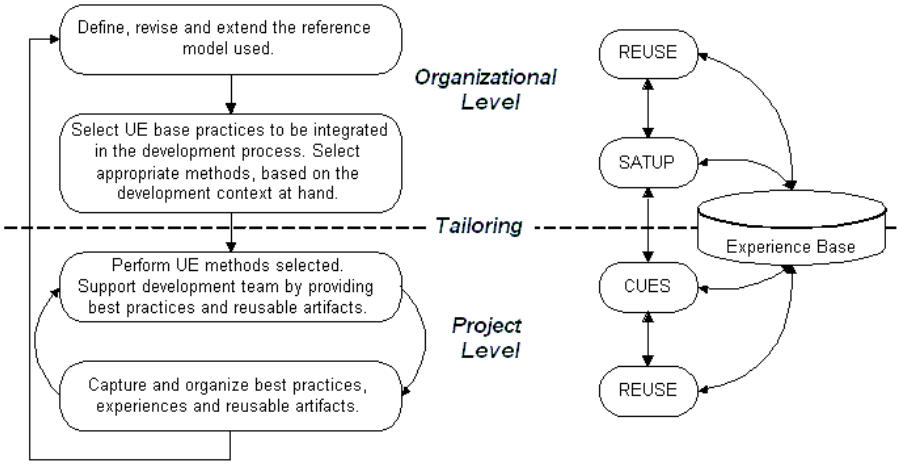
If a certain method was not useful to the development team, the team members can assign a poor rating to the respective USEPACK. This will reduce the likelihood for the USEPACK to be selected in subsequent projects with the same context situation. On the other hand team members can assign high ratings to USEPACKs which had a positive impact on their work thus increasing the likelihood that the respective USEPACK will be reused in subsequent projects. By applying the best practices described in a USEPACK members of the development team will make valuable experiences. These experiences can be captured as extensions or comments of an existing USEPACK or can be captured in a new USEPACK.

Assume that the development team used the USEPACK "Contextual Task Analysis" to perform the contextual task activity. Though they found the general guideline useful, they soon discovered that the contextual task analysis for designing an intelligent avatar is fundamentally different from the general method they found in the USEPACK. To capture the modifications to the general method they create a new USEPACK 'Contextual Task Analysis for Intelligent Assistants' using REUSE. They enter their experiences in the USEPACK and attach documents and templates that can be reused when performing a contextual task analysis for an intelligent assistant.

The next time a project team has to design an intelligent assistant, it will have access to the improved contextual task analysis method and the reusable artifacts. These improvement activities will prevent subsequent development teams to encounter similar problems and will enable them to perform the task more efficiently.

How the tool support provided by ProUSE is related to our evidence-based usability engineering approach is depicted in Figure 3.

Figure 3 : Tool Support for Evidence-Based Usability Engineering



6. EVALUATION

The goal of the evaluation was to get initial feedback on the validity of the underlying concepts and acceptance of the support tool ProUSE by development teams.

6.1 Test Subjects and Procedure

In this evaluation twelve employees of five large to small size companies[2] took part that are developing interactive software systems in domains such as home entertainment and telematics. The experiments were designed as semi-structured interviews. First, the general idea of the approach was presented and a short introduction to the ProUSE support tool was given to each participant. A questionnaire was handed out to collect personal data and data on the work experience of the interviewees. Each person was confronted with a scenario as described above and walked-though this scenario with the test supervisor. At defined points in the scenario the interviewees had to solve tasks such as defining a projects usability attributes or creating a new USEPACK. At the end of the interviews the subjects were asked if they would like to integrate the tool in their software development process and which modifications to the approach and the tool they would like.

6.2 Results

The evaluation of the pre-test questionnaire showed that the interviewees are largely involved in the implementation of the user interface, followed by requirements analysis and project preparation. Figure shows the degree of involvement of the interviewees in the various phases of the development

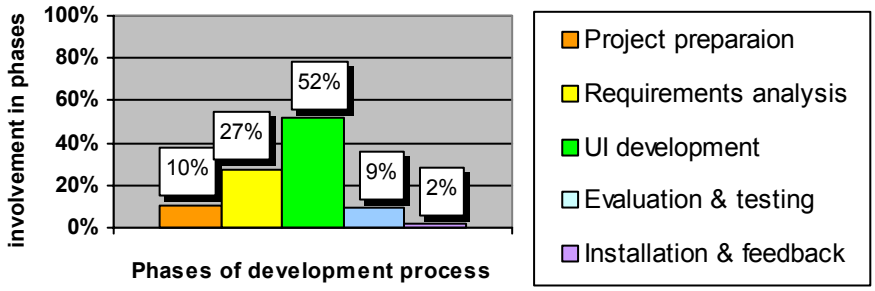


Figure 4 : Involvement of interviewees in phases of UI

process. Eight of the twelve developers interviewed would like to integrate the approach into their development process. Four interviewees would not integrate ProUSE into their development process without major modifications. As indicated by Figure 5, three of the four developers who rejected the approach had no or little experience in the development of user interfaces.

[2] Siemens (Munich), Sony (Stuttgart), Grundig (Nuernberg), Loewe (Kronach) in Germany

The votes for the evidence-based usability engineering approach might be linked to the long experience of the respective interviewees. They appreciate the idea to reuse existing HCD knowledge and methods for recurring design tasks. The rejection of the approach by less experienced developers suggests, that we should emphasize the benefits that new employees get from accessing and reusing an existing body of HCD knowledge.

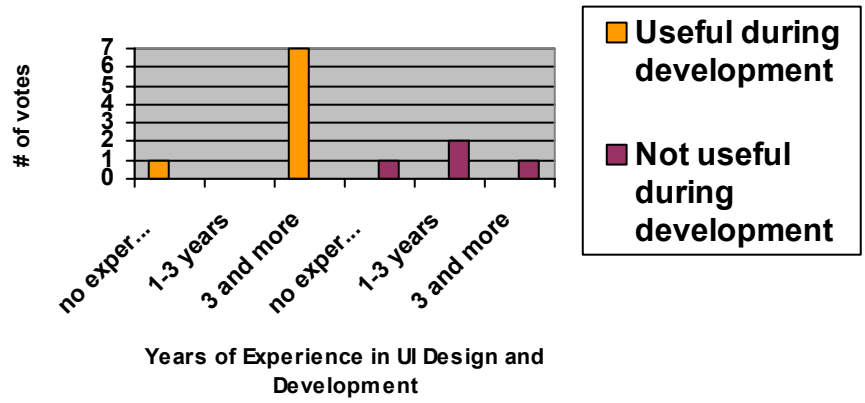


Figure 5 : Votes for deployment of ProUSE in development process

7. RELATED WORK

The *GUIDE* methodology [4] uses a similar approach to capture project experiences in a repository. So called usability resources consisting of interface patterns and a hierarchical structure of guidelines are used to represent the design knowledge. Context-specific instances of usability resources are called cases. A rule-based system is then used to match project characteristics to specific usability resources. The result is a set of usability guidelines that are assigned to the project. A review process is used to inspect whether the assigned resources are appropriate for the project in question. If there is a mismatch between project needs and the resources assigned, reviewers can recommend that either a different assignment is chosen or that the knowledge in the repository needs to be updated. Our reviewing process is based on the ratings of the project members as described in the meta-model in section 4.1. These ratings could be incorporated in an explicit reviewing process but ideally it would be part of the use of the ProUSE system. So in ProUSE there is no need for explicitly developing rules for the selection of knowledge. In ProUSE only the most efficient USEPACKs ‘survive’ the ratings by the development teams.

Following the ProUSE process optimal HCD activities 'emerge' from the experience base.

A main difference between ProUSE and GUIDE is that GUIDE uses a case-based architecture. With the help of the rule system the context of use of the design knowledge can be explicitly formalized. The ProUSE approach uses a semiformal model of the application context of a USEPACK and integrated assessments of the quality of a USEPACK to guide the USEPACK selection process. Another difference between the two approaches is that USEPACKs are oriented on HCD activities, whereas Cases are focused on a hierarchical structure of guidelines.

8. ACKNOWLEDGEMENTS

This research was sponsored in part by the BMBF[3] award #01 IL 904 B7 of the EMBASSI[4] project. We would like to thank all participants of our studies.

9. REFERENCES

1. Mayhew, D.J., *The Usability Engineering Lifecycle: A Practioner's Handbook for User Interface Design*. 1999: Morgan Kaufman Publishers.
2. Earthy, J., *Human Centred Processes, their Maturity and their Improvement.* IFIP TC.13 International Conference on Human-Computer Interaction (INTERACT), 1999. 2: pp. 117-118, http://info.lboro.ac.uk/research/husat/eusc/guide/tr_ump_c.doc.
3. Welie, M.v. *Breaking Down Usability*. in IFIP TC.13 International Conference on Human-Computer Interaction (INTERACT). 1999. Endinburgh, UK: IOS Press: pp. 613-620.
4. Henninger, S., *A Methodology and Tools for Applying Context-Specific Usability Guidelines to Interface Design.* Interacting with Computers, 2000. 12(3): pp. 225-243, .
5. Metzker, E. and M. Offergeld. *An Interdisciplinary Approach for Successfully Integrating Human-Centered Design Methods Into Development Processes Practiced by Industrial Software Development Organizations*. in IFIP TC2/TC13 WG2.7/WG13.4 Eighth IFIP

[3] Bundesministerium für Bildung und Forschung: German Ministry for Education and Research

[4] Elektronische, Multimediale Bedien- und Service Assistenz: Electronic, Multimedia Operating and Service Assistance

Conference on Engineering for Human Computer Interaction (EHCI'01). 2001. Toronto, Canada: Springer: pp. 21-36.

6. Metzker, E. and M. Offergeld. *REUSE: Computer-Aided Improvement of Human-Centered Design Processes*. in Mensch und Computer, 1. Fachübergreifende Konferenz des German Chapter of the ACM , MC2001. 2001. Bad Honnef, Germany: Teubner Verlag: pp. 375-384.
7. Norman, D.A., *The Invisible Computer*. 1998: MIT Press.
8. Nielsen, J., *Usability Engineering*. 1994: Morgan Kaufman Publishers.
9. Constantine, L.L. and L.A.D. Lockwood, *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. 1999: Addison-Wesley.
10. Beyer, H. and K. Holtzblatt, *Contextual Design: Defining Customer-Centered Systems*. 1998: Morgan Kaufmann Publishers.
11. Weinschenk, S. and S.C. Yeo, *Guidelines for Enterprise-wide GUI design*. 1995, New York: Wiley.
12. Billingsley, P.A., *Starting from Scratch: Building a Usability Programm at Union Pacific Railroad.* Interactions, 1995. 2(4): pp. 27-30
13. Spencer, R. *The Streamlined Cognitive Walkthrough Method: Working Around Social Constraints Encountered in a Software Development Company*. in Conference on Human Factors in Computing Systems (CHI'00). 2000. The Hague: ACM Press: pp. 353-359.
14. Rosenbaum, S., J.A. Rohn, and J. Humburg. *A Toolkit for Startegic Usability: Results from Workshops, Panels and Surveys*. in Conference on Human Factors in Computing Systems (CHI'00). 2000. The Hague, Netherlands: ACM press: pp. 337-344.
15. Basili, V.R., G. Caldiera, and H.D. Rombach, *Experience Factory*, in *Encyclopedia of Software Engineering*, J.J. Marciniak, Editor. 1994, John Wiley & Sons: New York. pp. 528-532.
16. Vanderdonckt, J., *Development Milestones Towards a Tool for Working with Guidelines.* Interacting with Computers, 1999. 12(2): pp. 81-118
17. Welie, M.v. and H. Traetteberg. *Interaction Patterns in User Interfaces*. in 7th. Pattern Languages of Programs Conference. 2000. Monticello, Illinois, USA: Washington University Technical Report number: wucs-00-29.
18. Oppermann, R. and H. Reiterer, *Software evaluation using the 9241 evaluator.* . Behaviour & Information Technology, 1997. 16(4): pp. 232-245
19. Sutcliffe, A., *On the Effective Use and Reuse of HCI Knowledge.* ACM Transactions on Computer-Human Interaction, 2001. 7(2): pp. 197-221