First publ. in: Revue d'Interaction Homme-Machine (2004), pp. 61-64

Integrating Usability Engineering Methods into Existing Software Development Processes via Evidence-Based Usability Engineering

Eduard Metzker[1] and Harald Reiterer[2]

[1] DaimlerChrysler Research and Technology Centre, Software Technology Lab, HCI Research Group, Ulm, Germany
eduard.metzker@daimlerchrysler.com,
[2] University of Konstanz, Department of Computer and Information Science, 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 more strongly preferred by developers with 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 tomorrow's 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 (Mayhew, 1999). While approaches such as the usability maturity model (UMM) (Earthy, 1999) 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 (Welie, 1999). 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 on 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 (Henninger, 2000).

This article is an elaborated version of the paper presented at the 4th international CADUI'2002 Conference, Valenciennes, France (Metzker and Reiterer, 2002). It is organized as follows. In this paper we propose six claims that point out shortcomings of current HCD practice and requirements for improvement. First we take a look at existing HCD process models and existing approaches to capture HCI design knowledge, trying to identify the organizational obstacles that hamper the establishment of these models and the use of the knowledge in mainstream software development processes. Parts of the claims are based on a survey we have made, which examined how exactly HCD methods are applied in real projects. To address the shortcomings and to meet the requirements described in our claims we present next an evidence-based usability engineering approach to the improvement of HCD processes. We show a first prototype of a computer-aided usability engineering environment called ProUSE, that we developed to support our evidence-based usability engineering approach. We discuss the deployment of ProUSE in an industrial setting and present results of an evaluation, that has been undertaken to get initial feedback on the validity of the underlying

concepts and acceptance of the support tool ProUSE by development teams. A short overview about related work and a discussion of our evidence-based usability engineering approach finishes this article.

2 Existing HCD process models: Theory and Practice

A review of 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 (Metzker and Offergeld, 2001a). We summarized these problems in the following three claims.

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 (Mayhew, 1999). A first attempt of an interesting hci-enriched process model for supporting human-machine system design and evaluation can be found in (Kolski and Loslever, 1998, Kolski, 1998).

Claim 2 *Most of the HCD approaches do not cover a strategy of 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 (Metzker and Offergeld, 2001b). Therefore HCD methods often can not be utilized because the necessary knowledge is not available within the development teams (Earthy, 1999, Mayhew, 1999).
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 Usability Engineering (UE) Methods into mainstream software development process must be understood as an organisational 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 (Norman, 1998). 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 the design of interactive systems in a variety of domains (DaimlerChrysler Aerospace (DASA) Ulm, Sony Fellbach, Grundig Fuerth and DaimlerChrysler Sindelfingen. All sites are located in Germany.) (Metzker and Offergeld, 2001b). The study revealed that the organizations examined are practicing highly diverse individual development processes. However none of the UE development models proposed by (Mayhew, 1999, Nielsen, 1994, Lim and Long, 1994, Constantine and Lockwood, 1999, Beyer and Holtzblatt, 1998) are actually used. Furthermore, the people 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. Few of the participants were familiar with methods like *user profile analysis* or *cognitive walkthroughs* (Polson, 1992) 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 an 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 the claims below:

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 lack 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 (Weinschenk and Yeo, 1995, Billingsley, 1995, Spencer, 2000, Rosenbaum et al., 2000) supports the observation that such usability best practices and heuristics are, in fact, compiled and used by software development organizations. Spencer (Spencer, 2000), 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. From experiences collected in the field of software engineering (Basili et al., 1994) it must be assumed that, in most cases, best practices like Spencer's ones 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, tools should enable software development organizations to explore which of the existing methods and process models of UE perform best 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 organisations to keep track of the application context of UE methods.

3 Existing 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. The general method of working with guidelines embraces two phases (Scapin, 1990):

- Phase1: Collection and organization. In this phase, human factors recommendations are extracted from existing resources. Attributes are assigned to each recommendation that allow accessing them in a database.
- Phase2: Structuring. In this phase, each recommendation is structured according to its premises and conclusions. Furthermore examples are added to each recommendation.

Important limitations of guideline based design and evaluation of user interfaces have already been described by Scapin (Scapin, 1990). He distinguishes between intrinsic problems of guidelines and usage problems. Intrinsic problems include incompleteness of guidelines when new devices, modes or metaphors emerge. This problem can only be solved through an empirical approach. Another intrinsic problem is the lack of specific answers to design problems. These answers are often not available. Finally many guidelines have been elicited in controlled experiments and often can not be transferred to other contexts. Usage problems are based on the finding that very difficult for non-human factors specialists to identify the meaningful keywords that may trigger human factors knowledge.

A detailed analysis of the validation, completeness and consistency of existing guideline collections has shown that there are a number of problems (Vanderdonckt, 1999), 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 these problems with guidelines, a different approach for capturing design knowledge has been developed, called interaction patterns. The concept of patterns was first described by (Alexander et al., 1977) to capture and communicate knowledge on architecture. Later this concept was popularized in the software engineering community (Gamma et al., 1993) and adopted in the filed of human-computer interaction (Tidwell, 1998, Nanard, 2002). 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 difference is that patterns make both the context and problem explicit and the solution is provided along with a rationale. Compared to guidelines, patterns 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 (Oppermann and Reiterer, 1997).

Another interesting approach to capture HCI design knowledge are claims (Sutcliffe, 2001). 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; and (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, and (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 (Earthy, 1999). According to these models, organizations are highly ranked on a maturity scale, if they are capable of tailoring 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 it to be saved in an experience base. The experience base allows developers 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 developers 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 (Earthy, 1999) 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 (Metzker and Offergeld, 2001a). 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 (Earthy, 1999) 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 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, such as templates for usability test questionnaires, results of conceptual task analysis or user interface mockups. These facilitate effective performance of the selected HCD methods.

- *Step 4: Disseminate, assess and collect 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 as depicted in figure 1: 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.

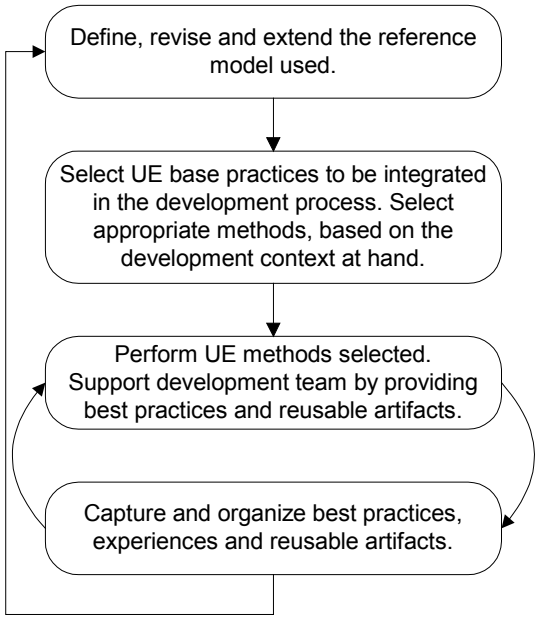

Figure 1. *Steps of the evidence-based model*

The outer cycle between step 4 and 1 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. The steps of the evidence-based meta model are depicted in Figure1.

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.

A USEPACK is structured into four logical sections:

- *Core information*
  The core information permits authors to describe the main message of a USEPACK. It is organized according to the pyramid principle for structuring information (Minto, 1987). The information first presented to the reader has a low level of complexity, allowing the reader to quickly decide if the USEPACK is worth further exploration. With further reading, the degree of complexity rises, introducing the reader to the experience described. The core information section includes the fields *title*, *keywords*, *abstract*, *description* and *comments*.

- C*ontext situation*
  The context situation describes the development context related to the experience in question. The context situation is generated by using the context model, allowing the authors and readers of USEPACKs to utilize a shared vocabulary for contextualizing and accessing USEPACKs.

- A*rtifacts and links*
  A set of *artifacts and links*. Artifacts such as checklists for user profile analysis or templates for usability test questionnaires, facilitate the efficient compliance with the best practice. They represent an added value to the readers of a USEPACK. Artifacts allow readers to regain time spent on exploring the package by using the supplied artifacts to simplify their work. Links are pointing to related resources that should be considered by the reader of a USEPACK.

- *Rating scheme*
  The *USEPACK rating scheme* allows engineering teams to assess the quality of a USEPACK along a defined set of dimensions such as 'ease of use' or 'quality of deliverables'. The ratings collected in projects are evaluated across project boundaries and together with the context situation form the rationale for selecting optimal USEPACKs for future projects.

4.3 A Computer-Aided Usability Engineering Environment

To increase the impact of our approach we developed a CAUSE (Computer-aided Usability Engineering Environment) 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 ProUSE prototype is based on Java technologies and integrated via a web portal concept which makes the modules available through intranet and web browser. Figure 2 shows the ProUSE Portal. After a user has logged into the system he or she can access the components of ProUSE that are dipicted in figure 2 in the blue circle.

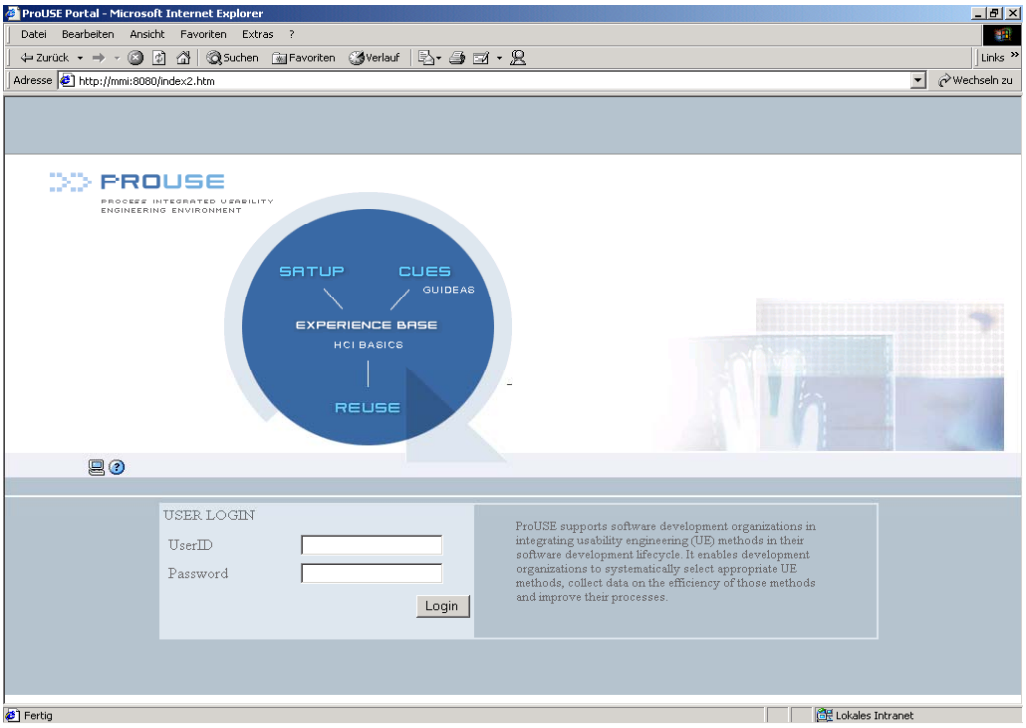

Figure 2. *Web-based access to the ProUSE portal*

The experience base is seeded with an initial set of best practices in the form of USEPACKs. In our case we have adopted a variety of usability engineering methods from Nielsen (Nielsen, 1994) and Mayhew (Mayhew, 1999) but in general any HCD approach should be appropriate. The REUSE (Repository for Usability Engineering Experience) (Metzker and Offergeld, 2001b) 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. Figure 3 shows how a USEPACK is edited with REUSE. On the left side, the user is provided with the textual description of the core information of a USEPACK. On the right side, the context situation for the USEPACK opened is visible. Other parts of a USEPACk can be accessed and edited via tab panes.

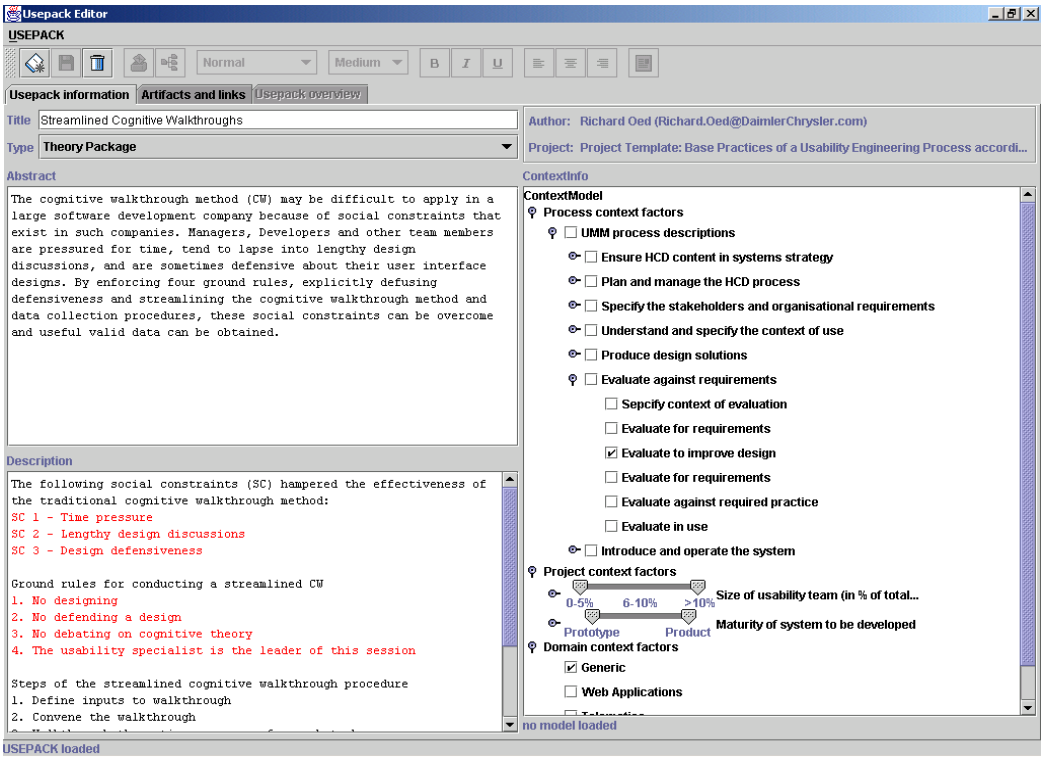

Figure 3. *Editing a USEPACK with REUSE*

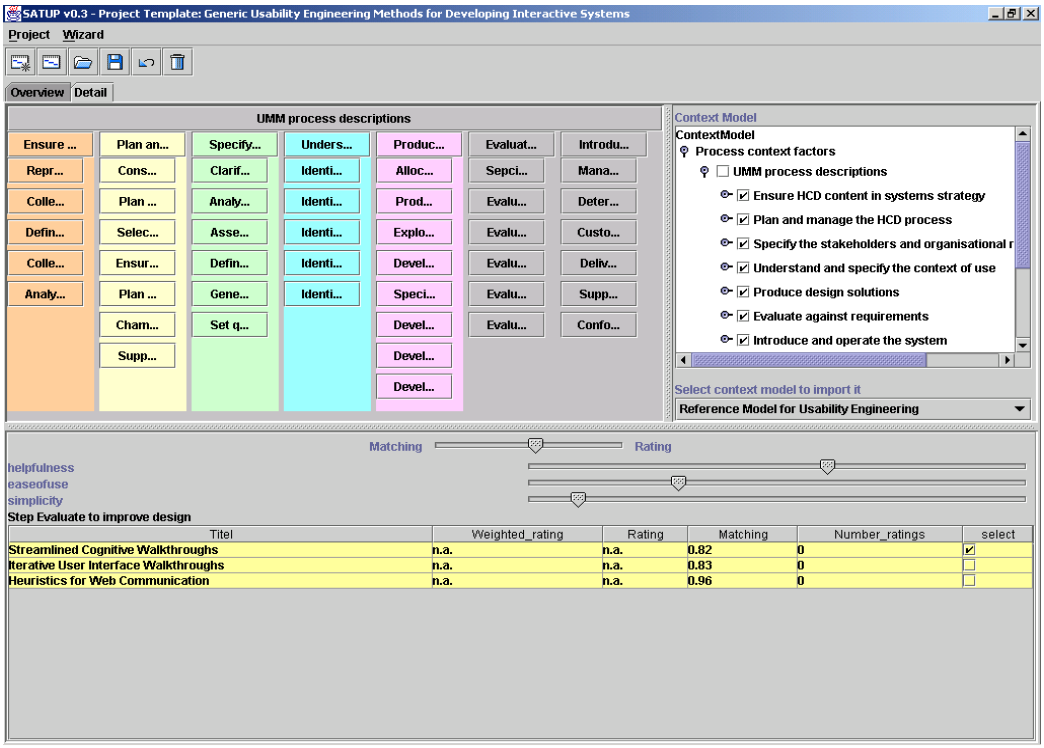

Figure 4. *Selecting appropriate USEPACKs with SATUP*

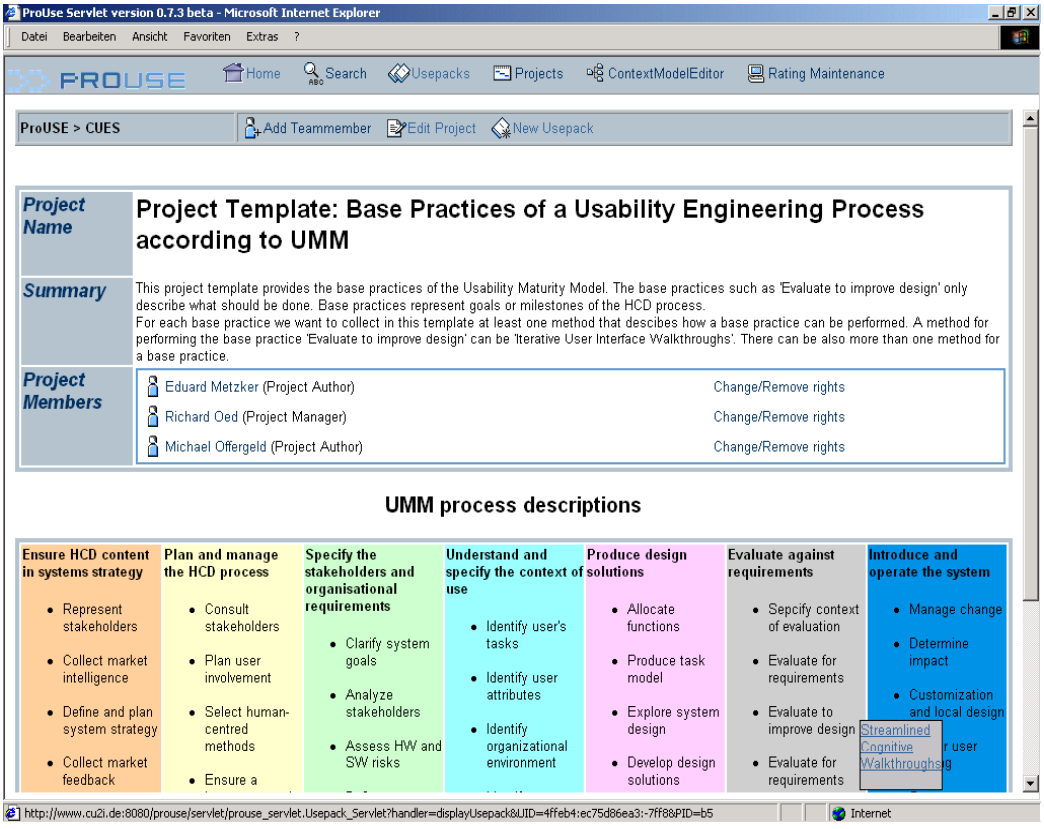

Figure 5. *Accessing USEPACKs via CUES*

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 (Kickert, 1978) and propose the optimal USEPACK.

Figure 4 shows the SATUP component. In the upper right corner, the project context situation is represented. It is used to describe important constraints for the selection of HCD methods for the project at hand, such as phases and steps of the reference model, resources available for HCD activities or the product maturity of the system to be developed. In the upper left corner, the phases and steps of the HCD reference model are listed. The user now iteratively walks through this model of the HCD process by clicking on each button that represents a process step. In the lower half of the screen the USEPACKs which have been found are depicted. In this area all USEPACKs are listed that match to the project's context situation. The similarity between the project's context situation and the context situation of a USEPACK is expressed via a matching factor. Furthermore existing ratings for a package are displayed. The list of USEPACKs can be sorted according to the rating factor, the matching factor or a combination between rating and matching. The package can be viewed by clicking on the USEPACK title. Finally one or more USEPACKs can be selected by clicking on the checkbox in the row of the respective USEPACK. This procedure is repeated for each process step of the reference model. Finally the project is saved and ProUSE constructs a hypermedia model of the reference model. In this hypermedia model each process step is linked with the USEPACKs that were selected for the step.

Once an optimal HCD process was planned with SATUP, CUES (Cooperative Usability Engineering Workspace) provides access to the hypermedia model of the HCD process and can be used by a distributed development team to access, perform and assess the HCD methods selected.

Figure 5 shows the CUES component. On the upper part, details of the project description and the team members that have access to this project are given. On the lower part of the screen you see the phases and steps of the HCD process. When the mouse pointer in navigated over a process step, the HCD methods that are linked to the process step are shown. In figure 5 the HCD process step 'Evaluate to improve design' is linked to the method 'Streamlined cognitive walkthroughs'. By clicking on the link the USEPACK that describes the 'Streamlined cognitive walkthrough' method is made visible.

In figure 6 the USEPACK 'Streamlined cognitive walkthroughs' is depicted. On the left side, a textual description of the USEPACK is displayed, together with comments of users of the USEPACK and attached artifacts. On the right side, the system shows the context situation of the USEPACK that describes under which constraints the USEPACK should be applied.

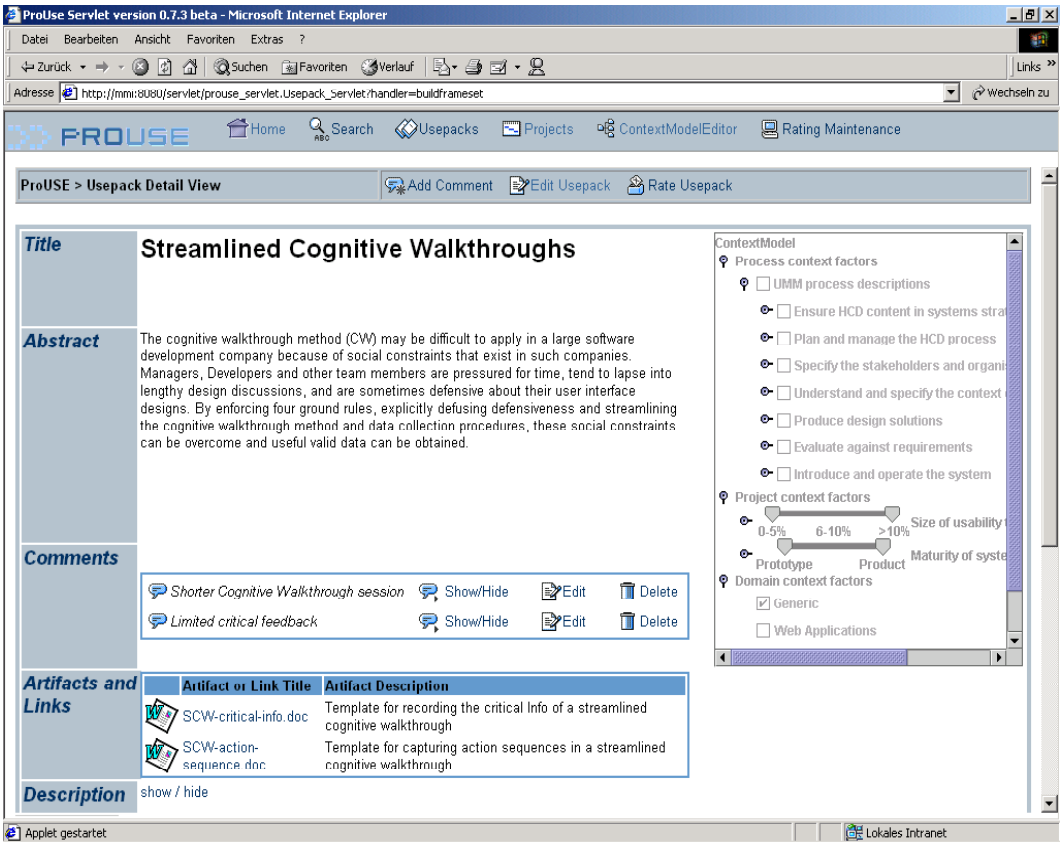

Figure 6. *View of a USEPACK*

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 project's usability constraints to the context factors of the context model thereby creating a context situation for the project. After the usability engineer modelled 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 being 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 being 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 from encountering 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 7.

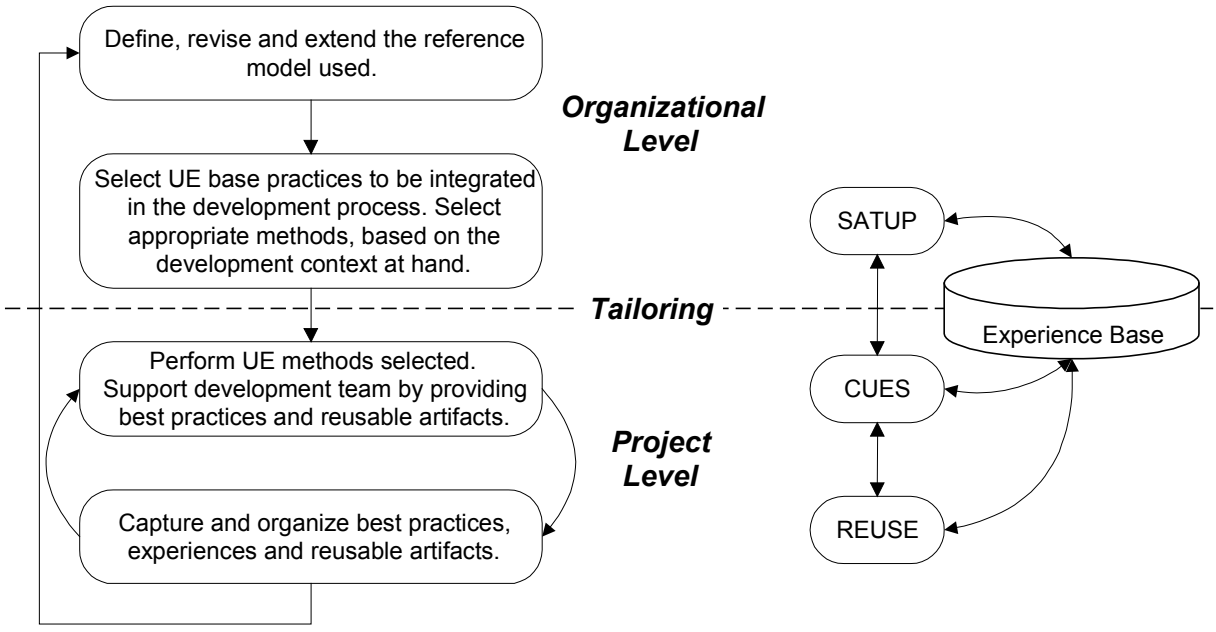

Figure 7. *Tool Support for Evidence-Based Usability Engineering*

6 Evaluation of ProUSE

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

Twelve employees of five four large to small size companies (Siemens (Munich), Sony (Stuttgart), Grundig (Nuernberg), Loewe (Kronach) in Germany) who are developing interactive software systems in domains such as home entertainment and telematics took part in this evaluation. 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-through 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 development of the user interface, followed by requirements analysis and project preparation. Figure 8 shows the degree of involvement of the interviewees in the various phases of the development 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 in Figure 9, three of the four developers who rejected the approach had no or little experience in the development of user interfaces.

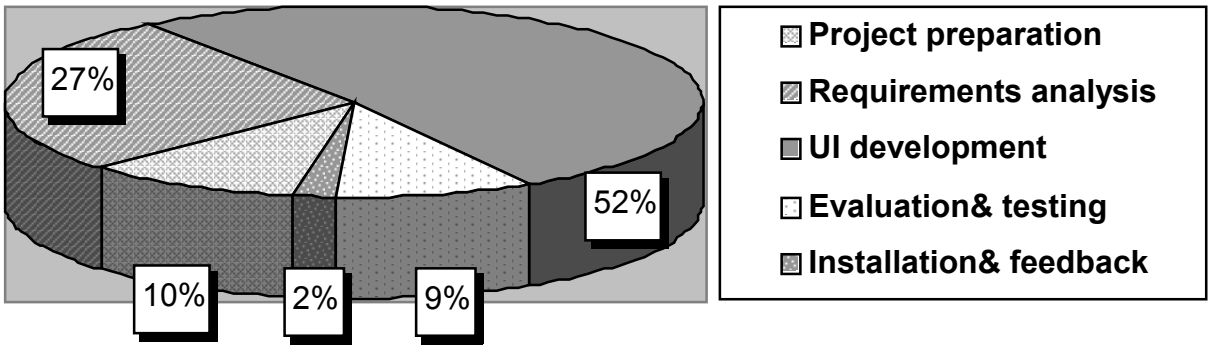

Figure 8. *Involvement of interviewees in phases of UI development process*

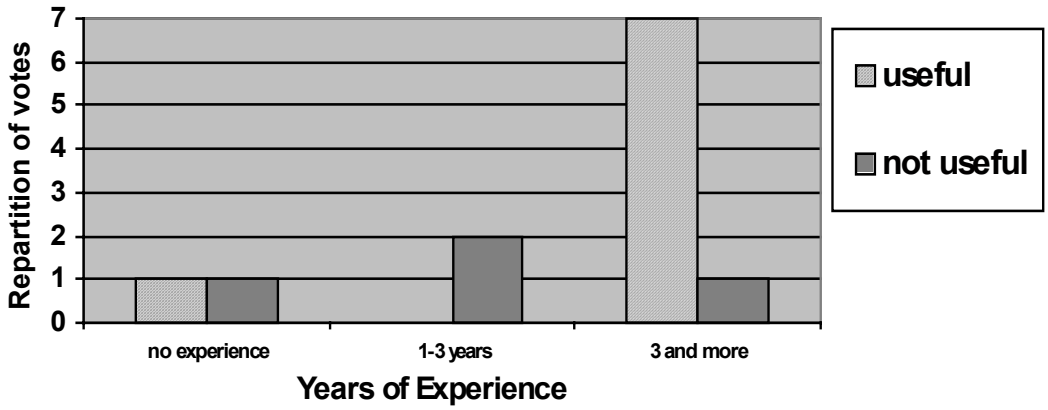

Figure 9. *Perceived usefulness of ProUSE during development depending on experience of subjects in UI design and development.*

The votes for the evidence-based usability engineering approach might be linked to the long experience of the respective interviewees. They appreciate the idea of reusing 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.

7 Related work

The *GUIDE* methodology (Henninger, 2000) uses a similar approach to capture project experiences in a repository. It is also a combination of tool and process to capture knowledge as it emerges in practice. 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 resources that are assigned to the project. The GUIDE methodology is embedded in an organizational learning approach. 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. These ratings could be incorporated in an explicit reviewing process but ideally it would be part of the use of the ProUSE system.

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 description with the help of the context model and a retrieval component based on integrated assessments to retrieve the best matching USEPACKs for a specific project context. Another difference between the two approaches is that USEPACKs are oriented on HCD activities. whereas cases are focused on a hierarchical structure of guidelines (Web-enhanced Smith and Mosier 944 guidelines corpus (Smith and Mosier, 1986)).

The User **I**nterface **D**esign **A**ssistant (IDA) (Reiterer, 1995) has inspired the development of the ProUSE system presented in this paper. The primary idea of IDA was also the explicit incorporation of HCI knowledge into the Usability Engineering process. This was done with the help of different design aid tools that have been integrated in a User Interface Management System (UIMS). The different design aid tools represent HCI knowledge (mainly guidelines but also specific methods for developing GUIs) in quite different forms (e.g. templates, hypermedia system, rules of a knowledge base) and assist usability engineers (primarily UI developers) during the design, implementation, and evaluation process of an UI.

The IDA *advice-giving tool* supports design and implementation activities presenting guidelines for GUIs with the help of an online hypermedia style guide. The access to the advice-giving tool could be context-sensitive or global. When the developer activates the advice-giving tool in a context-sensitive way, he gets the related guidelines of the selected interaction object (e.g., notebook) in the working area of the UIMS. When the developer activates the advice-giving tool in a *global way*, a graphical content browser of the online style guide appears. The online style guide offers a variety of possibilities to present the guidelines, e.g., text, pictures, screen shots, animations, and spoken explanations. The IDA *construction tool* supports implementation activities considering HCI knowledge with the help of a library of reusable ergonomic templates (e.g., controls, dialogues, windows). The templates in the library are not restricted to generic UI objects (e.g., push button, list box, table). The library also includes domain-specific window dialogues (e.g., primary window with different secondary windows and dialogue boxes) based on common window architectures. Using the library the developer generates an instance from each predefined template. The IDA *quality assurance tool* evaluates the ergonomic quality of GUIs with the help of a knowledge base representing the HCI knowledge in the form of production rules. When the developer demands an evaluation, the passive quality assurance (critique) is explicitly invoked. Using the rules of the knowledge base and controlled by an inference mechanism the quality assurance tool analyses the conformance of the UI with the human factors knowledge base (analytic critique). The results of the analytic critiquing process are presented in a special window. The window contains a short description of all discovered ergonomic deficiencies, the source of the short description and the identifier of the evaluated object that contains some ergonomic deficiency. The multifaceted architecture of IDA derives its power from the integration of its design aid tools. Therefore from each design aid tool, the other tools could be started in a context sensitive way.

The ProUSE system takes the IDA approach many steps further. It supports the whole usability engineering lifecycle and it offers with the help of the USEPACKs a much broader set of HCI knowledge (e.g. methods, processes, best practices, UI patterns, design artifacts). Another important difference is that the project context will be taken into consideration streamlining the methodological support (SATUP assistant). This streamlining process is not only based on the context factors, it also uses the USEPACKs ratings given by the usability engineers as a measurement. So the ProUSE is a consequent extension of the original IDA approach in many ways.

Another interesting approach supporting usability engineers is the *planning aid for evaluation practice* from (Denley and Long, 1997). The planning aid is intended to support novice usability engineers evaluate interactive software systems. For this purpose the planning aid offers first a specific component specifying the evaluation context. This component has much in common with our SATUP component specifying the project context. It allows specifying an evaluation plan considering different organizational factors (e.g. project status, competence, management issues) and performance factors (e.g. actual and desired quality of the product, costs). The result of this first phase is a document specifying the desired performance of the evaluation. It lists and prioritizes all factors that determine the desired quality and the acceptable costs of the evaluation. It includes also the goals and aims of the evaluation, and the criteria for success. This document is iterated with the client and can be agreed and signed off. Based on this document in the next phase the planning aid supports the usability engineers selecting the right evaluation method for the purposed evaluation setting. Like the Experience Base of ProUSE the planning aid includes a collection of possible evaluation methods with a detailed description of each method. All methods have been classified in three different classes: craft methods, applied science methods, and engineering methods. During the selection process the usability engineers are supported by a set of

heuristics (rules), to help them select between alternative methods, and resolve conflicts in the possible choice of methods. These rules are tacit or implicit knowledge that expert evaluators acquired through their work. The rules are represented in production rules (IF/THEN rules, with an additional BECAUSE statement to present a rational for considering the rule). This is an important difference to the ProUSE approach presented in this paper. Our approach is based on ratings of the different USEPACKs using sophisticated statistical methods for offering with the help of the CUES component the right USEPACKs for a specific project context. In a third phase the planning aid offers the possibility to configure the selected methods, to meet the desired quality and acceptable costs of the evaluation as specified in the first phase. For this purpose rules are also used and the result will be an evaluation plan. This evaluation plan consists of very detailed descriptions of each proposed evaluation method. This allows especially novice usability engineers to prepare an evaluation. The finale evaluation plan will guide them during all necessary evaluation steps. The planning aid offers no possibilities for the usability engineers to feed back experiences they made during the evaluation (e.g. ratings, comments about the usefulness, definition of new or customized methods). This is another important difference between our ProUSE system, where the usability engineer will be able to define or customize USEPACKs with the help of the REUSE editor and the planning aid tool. There is a hypermedia version of the planning aid available that supports the representation by providing a notepad facility that allows the evaluator incrementally to construct the plan an edit it as required. In comparison with our ProUSE system, the planning aid is focused only on the evaluation phase of a typical usability engineering lifecycle. It is therefore much more restricted in its methodological support.

In (Ella et al., 1999) a Decision Support System (DSS) for computer-aided choice of UI evaluation criteria and methods is presented. The aim of this system is also to support novice usability engineers selecting the right evaluation method considering different project constraints. One of the strength of this system is the comprehensive consideration of certain social and cultural differences concerning 212 countries and islands. The system called ADHESION is implemented in PrologIII using Fuzzy sets and production rules. In a first phase the user of the system has to answer different questions about the situational context (e.g. presence of representative users, stage in the development process, available financial resources, location) and about the potential performance of the evaluators (e.g. experience, current workload). Based on these different criteria the system automatically selects and proposes evaluation methods and criteria. The automatic selection process will be done with an inference machine using a knowledge base. The knowledge base models typical expert's preferences with the help of Fuzzy rules (e.g. IF the C_1 criterion "available financial resources" is 0.8 AND IF the C_2 criterion "presence of an expert in ergonomics" is 0 AND IF the C_3 criterion "dimension to be evaluated" = "usability" THEN the (a_j) action is "to adjust the human resources" is 0.9). This approach is very similar to the planning aid tool presented above. Therefore the similarities and differences with our ProUSE systems are more or less the same.

8 Discussion

Evidence-based usability engineering provides a multidisciplinary approach for integrating usability engineering activities into software development processes. In fact the approach has its foundations in the intersection of three disciplines: usability engineering, software process improvement and organizational learning as indicated in figure 10.

This interdisciplinarity is directly caused by our practical goal: to support the introduction, establishment and

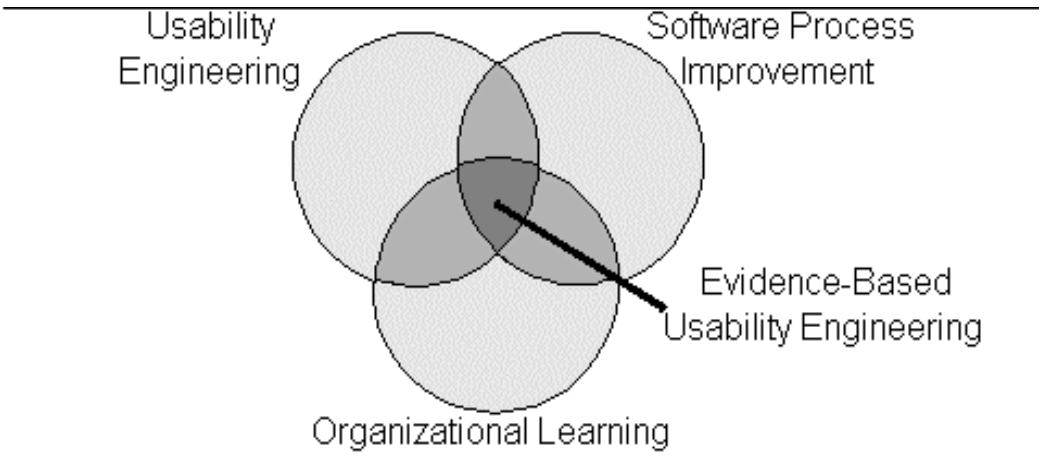

Figure 10. *Origins of the Evidence-Based Usability Engineering Approach*

continuous improvement of usability engineering activities in software engineering processes. We want to integrate usability engineering activities into software development processes to improve the engineering process for interactive software systems. As a tool to achieve process improvement we use concepts of organizational learning. Due to its multidisciplinary nature, our approach has some implications on each of the fields mentioned above. We will explore these implications in the following sections.

8.1 Increasing an Organizations Usability Maturity by Evidence-Based Usability Engineering

Recently the idea of process maturity was introduced to the field of usability engineering (ISO/TC 159 Ergonomics, 1999, DATech Frankfurt/Main, 2001). The rationale behind usability maturity models is that there are direct dependencies between an organization's capability to tailor its HCD processes according to various constraints, the quality of the development process performed and the quality in use of the interactive software system. Generally an organization has reached the maximum usability maturity level when it is able to perform, control, tailor and continuously improve its usability engineering activities. Our approach as discussed above is geared to support software engineering organizations in reaching high usability maturity levels. At a low usability maturity development organizations can use the experience base as a repository for base practices which are found in the initial set of USEPACKs (seed of the experience base) and which enable the engineering teams to perform those base practices. By applying the base practices in various projects, the development organization is able to derive USEPACKs for each base practice which capture project constraints such as 'size of development team' or 'available budget'. Thus the organization develops a set of more contextualized methods for each base practice, which will enable the organization to control its usability engineering activities. That means it can select the appropriate method for a given resource constraint from the set of USEPACKs available in the experience base.

By applying these controlled methods in future projects the organization collects experience on how to apply the available methods in different domains or for different technologies. By capturing these experiences in more contextualized USEPACKs and feeding them into the experience base the organization will increase its capability to tailor the set of available methods to a development context at hand. It will for example be able to have an optimized set of usability engineering methods for developing intelligent assistants with the avatar technology using a medium size budget with a small development team. By evaluating and eventually refining each method after applying it a continuous improvement process is triggered.

8.2 Balancing the Need for Structured Approaches and Creativity in Usability Engineering

It has often been claimed that there is a gap between the need for structured approaches in HCD on the one side and creativity in the process on the other side. The evidence-based usability engineering approaches tries to balance these needs. It does not force development organizations to run a second usability engineering lifecycle in parallel to the actual development lifecycle nor does it force engineering organizations to discard their established processes and replace them by a completely usability centered approach. Both of these directions have proved to be infeasible in practice. Instead evidence-based usability engineering provides a framework for smoothly integrating usability engineering activities into established software engineering processes. Following our approach usability engineering activities are not linked together in a fixed process model. Instead they are arranged in a construction-set-like reference model linked by a context model. The reference model and its base practices describe what should be done. The best practices together with their context situation describe how it should be done in a defined context situation. Engineering teams can select, evaluate, refine and extend base practices to meet their needs. By continuously evolving base practices to contextualized best practices (USEPACKs) engineering organizations compile a set of optimized methods which they can dynamically link into their development process.

9 Conclusion

In this article we presented an approach for integrating usability engineering process knowledge into existing software development processes: evidence-based usability engineering. Our approach was motivated by the discrepancy that we discovered between the overwhelming number of elaborated usability engineering approaches and the degree to which they are practiced in industry. Even usability engineering methods such as cognitive walkthroughs (Polson, 1992), which are regarded by the human-computer interaction community as being essential, are not known by development teams and are consequently not performed during the development of interactive software systems.

From our point of view the central problem of developing interactive software systems with high quality in use is not the lack of a 'silver bullet' (Brooks, 1995) methodology or tool which will be applicable to all development contexts. Instead the still persisting problem is the gap between the disciplines of software engineering and usability engineering in practice. Bridging this gap in software development organizations is a problem of organizational learning. To foster this organizational learning process we presented a meta-model, called evidence-based usability engineering. Knowledge about how to perform usability engineering activities for an engineering problem at hand must be disseminated, evaluated and improved in software development organizations in a way that makes these methods easy to use and efficiently applicable for development teams. As a form for capturing and processing UE process knowledge we suggested usability engineering experience packages (USEPACKs). We are currently using this structure to capture usability engineering best practices from finished projects and from a large e-business project that is still running. These projects represent first case studies for examining the capability of USEPACKs and context models for modeling usability engineering process knowledge.

We explained why development organizations should be provided with means to collect empirical data on the quality of USEPACKs that allows to relate project charactaristics, with usability engineering methods, experiences made by development teams and reusable deliverables. We use quality assessments performed by project teams to get feedback on the utility of the USEPACKs that have been applied in a project. These ratings are used to adapt the context situation of USEPACKs. If a USEPACK *U* was applied in a project *P* and got weak ratings by the project team of *P*, the context situation of *U* is adapted so that the probability is decreasing that *U* is proposed by ProUSE for subsequent projects with similar characteristics as *P*. Our assumption is, that the context situation of *U* will converge until it exactly describes

under which conditions *U* can be successfully applied. The implicit assumption is, that the quality assessments given for *U* by project teams in projects with similar characteristics are sufficiently consistent. We are planning to study this point in a controlled experiment with software developers. The goal of the study will be to find out, how rating scales for USEPACKs should be designed to ensure sufficient consistency of quality assessments between project teams.

We can also understand evidence-based usability engineering as a software process improvement approach. A central success factor for such approaches is the commitment of all people involved. The study described in section 6 is only a starting point for examining the acceptance of the approach by project teams. We are planning to perform extensive studies to examine the usability of our concepts such as USEPACKs and context models as well as the usability of the support tool ProUSE.

10 Acknowledgements

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

11 References

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, S. and Angel, S. (1977), *A pattern language,* Oxford University Press, New York.

Basili, V. R., Caldiera, G. and Rombach, H. D. (1994) *Experience Factory,*In *Encyclopedia of Software Engineering*, Vol. 1 (Ed, Marciniak, J. J.) John Wiley & Sons, New York, pp. 528-532.

Beyer, H. and Holtzblatt, K. (1998), *Contextual Design: Defining Customer-Centered Systems,* Morgan Kaufmann Publishers.

Billingsley, P. A. (1995), *Starting from Scratch: Building a Usability Programm at Union Pacific Railroad, Interactions,* **2,** 27-30.

Brooks, F. P. (1995), *The Mythical Man-Month: Essays on Software Engineering,* Addison-Wesley Longman Inc.

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

DATech Frankfurt/Main (2001) In *DATech Prüfbaustein: Qualität des Usability Engineering Prozesses*, Vol. 2001 Deutsche Akkreditierungsstelle Technik e.V., .

Denley, I. and Long, J. (1997), *A planning aid for human factors evaluation practice, Behaviour & Information Technology,* **16,** 203-219.

Earthy, J. (1999), *Human Centred Processes, their Maturity and their Improvement, IFIP TC.13 International Conference on Human-Computer Interaction (INTERACT),* Volume **2,** 117-118.

Ella, A. N., Kolski, C., Wawak, F., Jacques, C. and Yim, P. (1999), *An Approach of Computer-Aided Choice of UI Evaluation Criteria and methods.* in Computer-Aided Design of User Interfaces CADUI. 1999. Université catholique de Louvain, Louvain-la-Neuve: Kluwer: pp. 319-329.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1993), *Design Patterns: Abstraction and Reuse of Object-Oriented Design.* in European Conference on Object-Oriented Programming ECOOP'93. 1993.

Henninger, S. (2000), *A Methodology and Tools for Applying Context-Specific Usability Guidelines to Interface Design, Interacting with Computers,* **12,** 225-243.

ISO/TC 159 Ergonomics (1999) In *Human-centered Design Processes for Interactive Systems*, Vol. ISO International Organization for Standardization, .

Kickert, W. J. M. (1978), *Fuzzy theories on decision-making,* Martinus Nijhoff Social Sciences Division, Leiden, Boston, London.

Kolski, C. (1998), *A 'Call for Answers' araound the proposition of an HCI-enriched model, ACM Sigsoft Software Engineering Notes,* **23,** 93-96.

Kolski, C. and Loslever, P. (1998), *An HCI-Enriched Model for Supporting Human-Machine Systems Design and Evaluation.* in IFAC- Man Machine Systems Conference. 1998. Kyoto, Japan.

Lim, K. Y. and Long, J. B. (1994) *The MUSE Method for Usability Engineering,*In Cambridge University Press, Cambridge, UK, pp. 355

Mayhew, D. J. (1999), *The Usability Engineering Lifecycle: A Practioner's Handbook for User Interface Design,* Morgan Kaufman Publishers.

Metzker, E. and Offergeld, M. (2001a), *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 Conference on Engineering for Human Computer Interaction (EHCI'01). 2001a. Toronto, Canada: Springer: pp. 21-36.

Metzker, E. and Offergeld, M. (2001b), *REUSE: Computer-Aided Improvement of Human-Centered Design Processes.* in Mensch und Computer, 1. Fachübergreifende Konferenz des German Chapter of the ACM, MC2001. Bad Honnef, Germany: Teubner Verlag: pp. 375-384.

Metzker, E. and Reiterer, H. (2002), *Evidence-based Usability Engineering.* in Computer-aided Design of User Interfaces (CADUI2002). 2002. Valenciennes, France: Kluwer Academic Publishers.

Minto, B. (1987), *The Pyramid Principle - Logic in Writing and Thinking,* Minto International Inc., London.

Nanard, J. (2002), Les patrons dans la conception et la realisation d'hypermedias, to appear in online journal RIHM (online: www.limsi.fr/rihm/) , **3,** 41-77.

Nielsen, J. (1994), *Usability Engineering,* Morgan Kaufman Publishers.

Norman, D. A. (1998), *The Invisible Computer,* MIT Press.

Oppermann, R. and Reiterer, H. (1997), *Software evaluation using the 9241 evaluator, . Behaviour & Information Technology,* **16,** 232-245.

Polson, P. G., Lewis, C.H., Rieman, J., Wharton, C. (1992), *Cognitive Walkthroughs: a method for theory-based evaluation of user interfaces, International Journal of Man-Machine Studies,* **36,** 741-773.

Reiterer, H. (1995), *IDA - a design environment for ergonomic user interfaces.* in IFIP Conf. on Human-Computer Interaction (INTERACT'95). 1995. Lillehammer: Chapman & Hall, London,: pp. 305-310.

Rosenbaum, S., Rohn, J. A. and Humburg, J. (2000), *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.

Scapin, D. L. (1990), *Organizing Human Factors Knowledge for the Evaluation and Design of Interfaces, International Journal of Human-Computer Interaction,* **2,** 203-229.

Smith, S. L. and Mosier, J. N. (1986) In *Guidelines for Designing User Interface Software*, Vol. The MITRE Coporation, Bedford.

Spencer, R. (2000), *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.

Sutcliffe, A. (2001), *On the Effective Use and Reuse of HCI Knowledge, ACM Transactions on Computer-Human Interaction,* **7,** 197-221.

Tidwell, J. (1998), *"Interaction design patterns"*. in Patterns Languages of Programs (PLoP '98), (online: http://www.mit.edu/~jtidwell/interaction_patterns.html). 1998. Allerton Conference Center, Urbana Champaign, IL.

Vanderdonckt, J. (1999), *Development Milestones Towards a Tool for Working with Guidelines, Interacting with Computers,* **12,** 81-118.

Weinschenk, S. and Yeo, S. C. (1995), *Guidelines for Enterprise-wide GUI design,* Wiley, New York.

Welie, M. v. (1999), *Breaking Down Usability*. in IFIP TC.13 International Conference on Human-Computer Interaction (INTERACT). 1999. Endinburgh, UK: IOS Press: pp. 613-620.