H. Reiterer & O. Deussen (Hrsg.): Mensch & Computer 2012
München: Oldenbourg Verlag, 2012, S. 33-42

Prototyping-based Usability-oriented Knowledge Systems Engineering

Martina Freiberg, Frank Puppe

Department of Artificial Intelligence and Applied Informatics, University of Würzburg

Abstract

Despite increasing industrial as well as research-based interest, knowledge (-based) systems (KS) still challenge developers by their complexity. However, existing KS development methodologies in most cases lack an appropriate integration of UI design and usability evaluation activities, due to their focus on knowledge base development. Regarding UI and interaction design, KS consequently still often are built in a rather ad-hoc manner, lacking both reusability of proven solutions and potentially valuable experimentation with alternative UI and interaction designs. In this paper, we present the tailored KS development tool ProKEt, that enables efficient, agile KS development while specifically focusing on UI/interaction design, and that moreover seamlessly integrates usability evaluation functionality.

1 Motivation

Knowledge(-based) systems (KS) have become more and more established in various contexts: Examples are fault diagnosis for technical devices–e.g., (Cebi et al. 2009)–or advanced medical applications–e.g., (Padma & Balasubramanie 2009). Due to their development costs–in terms of effort, money, and time– often regarding especially and foremost the creation of the underlying knowledge base, KS UIs are most often implemented in a rather ad-hoc manner; this not only results in monolithic, context-specific systems with few to no reusability but overall system usability is considered even less thoroughly enough. Yet, application contexts such as fault diagnosis, medical diagnosis or legal consultation are highly critical: Often either a lot of money or even human health/life depend on their flawless operation. Thus, we claim that not only development of a sound knowledge base is an essential key factor for successful KS, but that it is at least equally important to provide for a supportive UI that fosters a high, overall usability.

In this paper, we propose the prototyping and knowledge systems engineering tool *ProKEt* for web-based KS for addressing lastly mentioned issue: On the one hand, ProKEt allows for the efficient, agile development of both prototypical and productive KS, thereby supporting several default UI styles and system types out of the box; on the other hand, data collection

as well as analysis mechanisms are directly integrated for supporting straightforward (usability) evaluation and comparison of diverse system types or dedicated system features. ProKEt thus contributes to the field of KS development as it denotes an all in one tool solution to affordably design, explore, develop, and evaluate potential KS solutions.

Related Work

Regarding specifically an integrated tool for prototyping-based engineering of KS and their usability evaluation, to the best of our knowledge there exists no previous work to date. However, for KS development in general, there exist several tailored software tools–such as JavaDON (Tomic et al. 2006), or KnowWE (Baumeister et al. 2011)–and methodologies–e.g., CommonKADS (Schreiber et al. 2001), or the Agile Process Model (Baumeister 2004). Yet, those approaches mainly focus on *knowledge base development* not (thoroughly enough) considering UI/interaction design or integrated usability evaluation activities. In contrast, our proposed KS engineering tool ProKEt integrates efficient, rapid KS development with creative, experimental UI/interaction design and with usability evaluation activities. Therewith we assent to recent research that proved the value of user-centered development in terms of tightly combining (various) prototyping approaches and usability evaluation, see (Holzinger et al. 2011); named research does not specifically address the KS domain, yet the target system–a form of medical questionary–is basically quite similar to the *questionary style for documentation KS*, see previous work (Freiberg 2012). The integrated, agile development approach based on Extensible Prototyping (Freiberg 2012) specifically supported by ProKEt further roughly resembles the XU approach introduced by (Holzinger & Slany 2006)–a tight integration of Extreme Programming and Usability Engineering. (Leichtenstern & André 2010) further termed *user-centered prototyping tool* as an all-in-one tool solution that enables developers to efficiently, effectively and satisfactorily design, evaluate and analyze developed artifacts; in that sense, ProKEt can be seen as a user-centered prototyping tool for web-based KS. Concerning usability evaluation–specifically collecting logging data such as click logs–to date there exist a vast range of both research-based and commercial tools; however, they most often are separate tools that need to be installed, configured, or integrated with the website or program to evaluate. In contrast, prototypes and productive KS developed with ProKEt can be directly and seamlessly equipped with tailored evaluation mechanisms: They require no further installation, rendering their usage straightforward at any development point in time.

Paper Structure

The rest of the paper is organized as follows: In Section 2 we introduce the Prototyping and Knowledge Systems Engineering tool *ProKEt*. Afterwards, we present particularly the usability extension of ProKEt in more detail in Section 3. We then describe experiences from applying the tool in a current project in Section 4. Finally, we provide a discussion of the approach as well as a short summary of the presented work and an outlook to prospective future work in Section 5.

2 ProKEt

ProKEt is a tailored **Pro**totyping and **K**nowledge systems **E**ngineering **t**ool for web-based systems, that additionally provides support for various usability evaluation related activities. For a detailed introduction of ProKEt, and particularly the process of its supported agile, prototyping-based KS engineering process, see (Freiberg et al. 2012). The main application logic is implemented in Java; the resulting KS artifacts basically are Servlet-based web applications, using HTML, StringTemplate, and CSS for creating the UI, and integrating JavaScript for realizing the necessary interactivity. ProKEt is developed as an open source project and can be obtained from the web[1]. ProKEt further specifically supports consultation and documentation KS, see classification in (Baumeister 2004). A consultation system thereby provides decision support in a particular problem area based on the given user input, e.g., giving advice on specified legal topics. A documentation system contrastingly focuses on supporting uniform, efficient, high quality (regarding completeness and correctness) data entry. Due to the known advantages of web-based applications–e.g., maintainability, platform-independence, or user acceptance–and verified by own experiences with past KS projects, ProKEt further particularly focuses on web-based KS. For leveraging the process of engineering knowledge systems, ProKEt supports the creation of prototypical demo systems (pure prototypes) on the one hand, and the implementation of fully-fledged knowledge systems for productive use (productive systems) on the other hand. Thereby, the transition from prototype to productive system is possible nearly without additional effort by simply changing the specification of the data source: XML-based for prototypes and a *d3web*[2] knowledge base for productive systems. This is possible, as ProKEt uses the same base set of UI templates both for prototypical and productive artifacts. UI widgets are basically implemented with StringTemplate, rendering the addition of further widgets or the adaption/extension of existing widgets a straightforward task. For a more extensive introduction of ProKEt, the technologies used for prototypes and productive KS, and the straightforward, Extensible Prototyping approach, see previous work (Freiberg et al. 2012).

3 ProKEt: Usability-Extension

For the purpose of supporting several forms of usability evaluation, ProKEt offers the possibility to integrate both *quantitative* and *qualitative data collection* with both pure prototypes as well as with productive knowledge systems (KS); thus we refer to both types of ProKEt artifacts when speaking of *system* in the following. Thereby, usability features are activated by simply adding properties to the XML-based system specification.

---

1 http://proket.sourceforge.net/ (last checked Nov.3rd,2011).

2 http://d3web.sourceforge.net/ (last checked Nov.1st,2011).

Quantitative Measures

ProKEt provides for a tailored click logging mechanism that captures all relevant keyboard actions during KS usage. Those are on the one hand potentially interesting, global UI elements and actions, such as creating new-, saving-, or resuming sessions. On the other hand, all activities related to characteristic KS elements and interactions are logged along with the corresponding timestamp; an example is question answering, where the question, the provided answer, and a corresponding timestamp are recorded; for a detailed elaboration on KS-specific elements/interactions, see (Freiberg et al. 2012). In the case of consultation systems, furthermore the results of the problem solving session are logged. Based on the collected log-data, ProKEt automatically can derive several (usability) metrics, as introduced by various experts from the usability domain, e.g. (Bevan & Macleod 1994; Constantine & Lockwood 1999; Nielsen 1993). Examples are *Average Task Duration*, *Success Rate*, or *Number of Unused Widgets*. ProKEt allows for deriving such metrics both on a single-user basis, but also averaged over all users, as it often can provide additional insight to analyze certain metrics from both points of view: For example, an unusually high task duration value might indicate a major problem with the system as a whole, but might as well be due to problems of (some) specific user(s) thus producing severe outliers. By applying slightly tailored calculations, those metrics allow for a rough assessment and valuation of the general usability and effectiveness of a KS; an example for such tailoring is the definition of *success*, which in the context of consultation systems is defined as *correctly derived solution* in ProKEt. Further, we aim at detecting specific problems with particular UI widgets, e.g., by identifying unused/extensively used elements, or by investigating the extent of the help system usage for given UI elements or the entire system.

Qualitative Measures

ProKEt further enables the integration of (usability) questionnaires with its artifacts; those can be presented e.g. by offering an additional button/link, that opens the corresponding survey, or by displaying the questionnaire automatically right after the session has been completed by the user. Corresponding query forms are–as all UI widgets in ProKEt–also created and included based on the StringTemplate mechanism of the tool, thus they are easily exchangeable and adaptable with no effort for different needs. Currently, ProKEt provides the *System Usability Scale (SUS)* (Brooke 1996) and the *NASA Task Load Index (NASA TLX)* (Hart & Stavenland 2006) as default questionnaires, as well as a tailored questionnaire designed to specifically fit the needs of consultation KS evaluation. Additionally, a mechanism for providing *anytime feedback*–a means for collecting loose user feedback regarding, e.g., the system design or potential malfunctions at any point in time during system usage–is implemented.

ProKEt Usability-Extension: Use Cases and Benefits

Basically, all usability-related data–such as click log data, anytime feedback, or also questionnaire values–are gathered in one JSON-based text file; this can be further processed by ProKEt to either calculate basic usability metrics (see above) right away, or to just represent the collected data as CSV file, which in the further course can easily be imported to external

tools, such as standard spreadsheet programs or comprehensive statistical software, for more extensive evaluation.

The possibility to quickly and easily create both prototypical and productive KS with ProKEt and to effortlessly activate usability features for both types of artifacts provides strong support for *lab-setting* evaluations–such as user studies/experiments–at any time and independent from the concrete development state. Thereby, lab-setting evaluations become an affordable and attractive task to perform during development of a KS potentially even more than once. Furthermore, *field-setting* evaluation–by which we understand continuous evaluation of KS that are already in productive use–becomes straightforward due to the seamlessly integrated data collection mechanisms. Such evaluations can help to identify additional/other kinds of problems that might not occur in more strictly planned and conducted lab-based evaluations; an example is problems due not being able to instruct and guide users as extensively as in lab-setting evaluations. Furthermore, field-setting evaluation provides the chance to discover and to react to long-term effects of the system; for example, an elaborate consultation system, presenting always only one question at a time, might become annoying/boring to users with repeated usage and familiarization, whereas sophisticated systems that initially might appear complex and hard to use could turn out quite usable and efficient with increasing familiarization of the users. ProKEt therefore fosters usability for KS development in two ways: Explicitly, by strongly supporting specific experiments and studies, but also implicitly, by simply enabling a highly iterative, yet affordable, development process in the course of which potentially even more system assessments and refinements become possible. Another advantage of the proposed approach is the possibility, to not only evaluate the KS UI but also the knowledge base intuitively on a visual level–this is especially relevant, as KS UI and knowledge base are tightly coupled and changes on the one side, e.g. the knowledge base, can have severe effects on the other side.

4 JuriSearch: Web-based, Legal Consultation

JuriSearch was initiated at the beginning of 2012 as cooperation between the University of Würzburg and the RenoStar Corporation, partly founded by the Free State of Bavaria. JuriSearch aims at building a knowledge-based web consultation system for the legal domain, intended to provide quality advice on various legal topics, e.g., right of cancellation; target users range from legal laymen–searching for a basic estimation of their case–to junior staff lawyers seeking for guidance regarding legal (sub) domains that are not exactly their special field of work. The web-based system thereby intends to provide various different *clarifying consultation* modules, each of which assesses exactly one distinct core. In JuriSearch, ProKEt has been applied both for creating two initial prototypes of potential UIs and for conducting a first usability study for their comparison regarding their general applicability and usability using the integrated usability extension of the tool.

Figure 1 presents the two initial prototypes for the JuriSearch clarification consultation modules: A *one-question (One-Ques)* style (see Figure 1, A) and a *structured navigable-tree (Nav-Tree)* style (see Figure 1, B). The core issue to be rated is the question, whether a giv-

en dismissal was legally correct and thus effectually. Thereby, three final ratings–affirmed, undecided and rejected–can be derived. Both UI variants provide for a similar base interaction: The core issue is rated by assembling the rating of several questions; each such

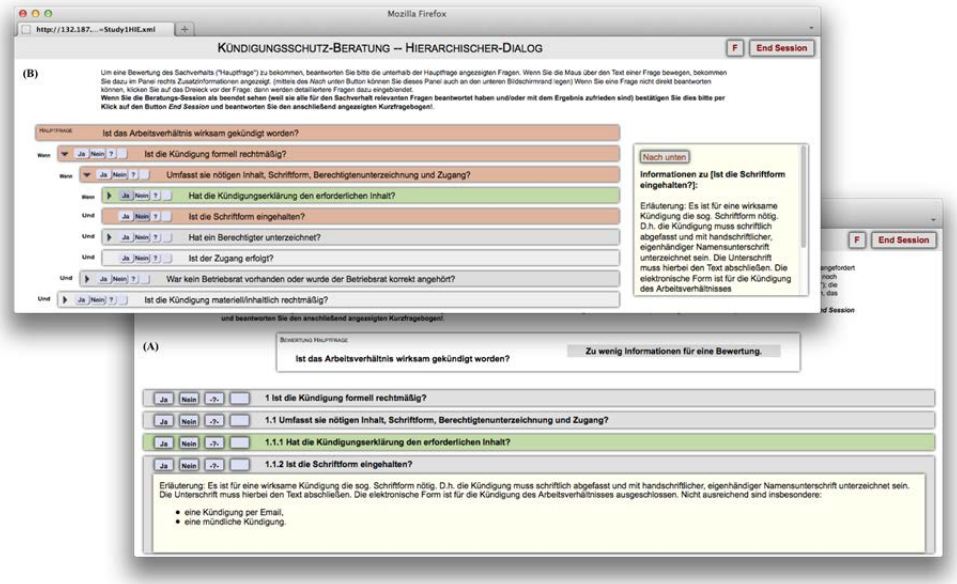

Figure 1: Two alternative prototypes for the JuriSearch consultation system UI/interaction style: One-question style (A) and structured navigable-tree style (B)–both in German.

question in turn can contain child-questions that further refine its parent element by querying in a more detailed manner. The questions generally provide answer alternatives *Yes*, *No*, and -?- *[Undecided]* and additionally offer the option to go into more detail by displaying refining questions. *One-Ques* thereby aims at imitating a strict conversation between the system and a user. Therefore, the system always presents only the one appropriate next question with answering options and additional explanations at a time; each question can either be answered directly, or be clarified on a more refined level by clicking the *Details* button–resulting in this case in displaying each of a question's refining elements, again in the one-at-a-time manner. Additionally, auxiliary information as e.g. more elaborate explanations helping to rate the current question is presented at the bottom of each question. The core issue to clarify is displayed at the top of the UI and is always immediately updated regarding its rating based on the current user input. The conversational interaction style is intended to ease system usage by helping the user focus always on the current question, not needing to think about the order of proceeding, as the system automatically guides the workflow by presenting the respective next suitable question based on the provided user input. Contrastingly, *Nav-Tree* presents all UI elements in an interactively navigable tree structure. Thereby, the core issue is displayed as the topmost element, displaying all questions required for its clarification as direct child-elements (branches); in case a user is not yet able to answer the currently expanded question, more refined detail questions can be expanded by clicking on the

arrow button in front of each question, whereas elaborate information for each question is displayed a designated information-box. By not only presenting the current next question but also the surrounding questions and hierarchical structure, the Nav-Tree UI style provides a kind of focus-and-context view. This is intended to foster an active exploration of the interface, and thereby to help the user gain expertise on the contemplated topic.

Method

The two systems were installed on a server for conducting a remote study, allowing participants to conduct their tasks when- and wherever they desired. Objective measures were collected using the built-in data logging mechanism, and subjective user results were queried with one of the short (computer-based) built-in questionnaires after each problem solving session; the questionnaire was in parts based on the SUS, yet containing several items intended to investigate some consultation KS specifics, such as the users believe in the system's final rating. Two exemplary problem descriptions from the legal domain were provided to the participants by email, along with the required instructional material. The participants were asked to solve one problem with Nav-Tree and one problem with One-Ques. For eliminating potential bias on the results due to the chosen UI sequence, the sequence was alternated between participants. In total 21 male research assistants and student staff members, mostly between 25 and 35 years, of the department were recruited. As computer science staff, they all had a high level of general computer experience, yet mostly no experiences regarding the legal consultation domain and the specific KS/UI types.

Results & Discussion

For assessing the *average task time*, the task time was measured as dependent variable. We found an average task time of 13m 38s (SD 6m 49s) for Nav-Tree, and of 10m 39s (SD 5m 49s) for One-Ques; by a narrow margin, this is statistically not significant (one-sided unpaired t-test, p=0,068). One possible explanation of Nav-Tree's higher average task time, supported by user feedback, is that Nav-Tree provided intuitive support for free, extensive exploration of the entire system. However, in our opinion the task time should not be overrated at all, here; the extent of usage and thus resulting task time of the test systems depends in larger parts on a) the reading speed of the participants regarding the questions and (often extensive legal) explanations, b) the usage conditions (during daily job routine or after end of work) which, due to the setting of the remote study could not be controlled strictly, and c) the potentially already existing knowledge on the matter of fact. This generally leads to highly participant-specific task time results, which makes the significance of the different task time values questionable in general. Yet, the tendency, that the average task time is not inordinately high in both cases can be valued as a positive sign. Finally, we also suspect a high learnability of Nav Tree, i.e. once users have familiarized with the UI, the efficiency of its usage could further increase, thus lowering the task time. This is to be investigated in separate, further studies.

Regarding the *Success/Error rate*, a case was rated successful, if the correct solution was derived by the user and not successful if either the wrong or no solution was found. For Nav-Tree, the Success Rate was 42,86% (one-sided binomial test, no statistical significance with p=0,11) and for One-Ques the Success Rate was 38,1% (one-sided binomial test, no statisti-

cal significance with p=0,16). This clearly indicates the need for overall system improvement, as those success rates are better than "just guessing"–where in this case, one could guess correctly by a 33% chance due to the three possible ratings of a core issue–yet in our opinion to be truly beneficial such a system should exhibit significantly higher success rates. Due to user feedback, we strongly suspect KB design factors (as opposed to UI design factors) such as the wording and sequence of the questions, to have influenced the usage of both systems equally negative, thus aggravating providing the correct answers and in turn also the derivation of the correct core issue rating.

Regarding the *UI Preference*, within the subjective questions, of the 21 participants 17 (~81%) preferred the Nav-Tree, 3 (~14%) preferred the One-Ques, and only 1 (~5%) stated no explicit UI preference. This is statistically significant with $p<0,05$ on a X^2 test; therefore, a distribution of 50% (Nav-Tree), 30% (One-Ques), and 20% (both equally) was anticipated due to our suspicion that Nav-Tree is a quite effective, intuitively usable UI representation especially regarding the study user group. The clear preference of Nav-Tree over One-Ques in the presented study may be due to the specific characteristics of the participants that–even not being expert users regarding the target domain–are more used to tree representations as computer scientists and thus might have perceived Nav-Tree as naturally intuitive. With regards to the further subjective questions, Nav-Tree scored better than One-Ques on every matter, using a scale from 0 (worst) to 6 (best): Comprehensibility of the system reactions 4.43±1.54 (Nav-Tree) vs. 2.76±1.45 (One-Ques) or of the derived results 4.53±1.54 (Nav-Tree) vs. 3.33±1.85 (One-Ques), the user's own estimation of whether (s)he could solve the case correctly 3,67±1,53 (Nav-Tree) vs. 2,52±1,83 (One-Ques), and the mediation of domain knowledge to the user 4.05±1.32 (Nav-Tree) vs. 2.95±1.72 (One-Ques); those differences are all statistically significant using an unpaired, one-sided t-test with $p<=0,05$. However, results could completely differ regarding other user types, e.g., with little web/computer experience as then a highly guiding UI as One-Ques potentially could score better; this is subject to further studies, though.

The *Anytime Feedback* mechanism further revealed valuable insights: Regarding the knowledge base, the wording of the questions/explanations was perceived as incomprehensible/complicated/too specialist in 11 cases (~52%), which probably highly influenced the overall study results negatively. This might have been aggravated by the fact, that the chosen participants were no specialists and thus not familiar with specific, legal terming and language; yet, as the target system is explicitly aimed at a diverse user population, a further knowledge base refinement with regards to clearness and understandability is indispensable. Based on that general comprehensibility problems with the knowledge base, further the idea arose to implement the entrance into the tree–i.e., the top-level questions directly rating the core issue–by questions highly important and understandable from the users' point of view, regarding for example potential reasons for dismissal. Finally, regarding particular UI design issues, 4 users (~19%) stated to be confused by the -?- button, originally intended as answer alternative *undecided*; they rather suspected more help/details regarding the current question to be displayed instead of causing just a question rating. Similarly, 3 users (~14%) stated to not have understood the *empty* button–designated to clearing a previously entered answer.

5 Conclusion

When developing for usability, iterative development is reputed essential, see e.g. (Nielsen 1993). In previous projects, ProKEt already turned out to be a powerful support for agile, iterative development of KS prototypes and productive systems: The *Mediastinitis* and the *EuraHS* project are reported in (Freiberg 2012). Regarding the most recent *JuriSearch* specifically, ProKEt not only enabled an efficient creation of the two initial prototypes, but also drastically eased the first, comparative user study by its built-in, tailored usability features, revealing extensive and valuable insights regarding the general usability but also regarding the most severe drawbacks of both alternatives. Regarding usability evaluation based on automated data collection and analysis in general, we are well aware that automated measures can hardly be sufficient for thoroughly evaluating any system's usability. Yet, the tailored measures nevertheless can provide valuable insight regarding major problems of/with the investigated KS UIs and furthermore drastically ease comparative assessments of different designs; additionally, the simultaneous collection of qualitative data can provide further understanding regarding the quantitative results. To strengthen the approach, we suggest to integrate other known usability techniques additionally–such as observation, think aloud, or interviews; this offers the advantage, to reveal even more insights and to make sense of otherwise potentially ambiguous data. We are also aware, that there already exist manifold approaches and tools for logging both mouse-based and keystroke-based activities. However, as we aimed at capturing particularly the activity flow during KS sessions, we implemented an own, tailored logging approach based on mouse-click- and keystroke-data that collects only exactly those information required for best supporting KS evaluation.

In this paper, we argued that both interaction/UI design and usability evaluation activities still are a rather unconsidered, but even the more important, issue when implementing knowledge systems (KS). To leverage that issue, we introduced the tailored KS development tool *ProKEt*. Current projects already showed the general applicability and value of the tool both as a means for efficiently developing KS as well as a tool for conducting seamlessly integrated usability studies. Prospective future work includes: Further studies regarding more diverse users (including, e.g., legal experts) as the future legal consultation system is intended to be used by a very diverse user population as well as studies for investigating the suspected learnability of the Nav-Tree UI. Regarding the tool ProKEt itself, even further usability metrics, automatically derived from the log files, could be added, such as metrics that compare actual performance with some benchmark values (e.g., *Productiveness* as introduced by (Bevan & Macleod 1994)). Further, ProKEt could profit from including mouse-movement tracking (extending the existing click logging) for gaining even more detailed insight regarding the actual interaction of users with the investigated KS UIs. Finally, we consider enhancing the tool by automatically created (visual) representations of at least the basic evaluation results/metrics.

Acknowledgement

We thank the RenoStar (Großwallstadt, Germany) staff members for valuable discussions while developing the first prototypes and for providing their specialist contents.

References

Baumeister, J. (2004). *Agile Development of Diagnostic Knowledge Systems*. IOS Press AKA DISKI 284.

Baumeister, J., Reutelshoefer, J. & Puppe, F. (2011). KnowWE: A Semantic Wiki for Knowledge Engineering, *Applied Intelligence*, *35*(3), 323–344.

Bevan, N. & Macleod, M. (1994). Usability Measurement in Context. *Behaviour and Information-Technology*, 13, 132–145.

Brooke, J. (1996). SUS: A quick and dirty usability scale. In Jordan, P.W., Weerdmeester, B., Thomas, A. & Mclelland, I. L. (Hrsg.): *Usability evaluation in industry*. London: Taylor and Francis.

Cebi, S., Celik, M., Kahraman, C. & Er, I. D. (2009). An expert system towards solving ship auxiliary machinery troubleshooting: shipamtsolver. *Expert Systems with Applications*, *36*(3), 7219 – 7227.

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

Freiberg, M., Striffler, A. & Puppe, F. (2012). Extensible Prototyping for Pragmatic Engineering of Knowledge-based Systems. *Expert Systems with Applications*, doi: 10.1016/j.eswa.2012.02.110.

Hart, S. G. (2006). Nasa-Task Load Index (Nasa-TLX); 20 Years Later. In: *Proceedings of the Human Factors and Ergonomics Society 50th Annual Meeting*. Santa Monica: HFES. [904–908].

Holzinger, A. & Slany, W. (2006). XP + UE -> XU Praktische Erfahrungen mit eXtreme Usability. *Informatik Spektrum*, *29*(2) 91-97.

Holzinger, A., Kosec, P., Schwantzer, G., Debevc, M., Frühwirth, J., Hofmann-Wellenhof, R. (2011). Design and Development of a Mobile Computer Application to Reengineer Workflows in the Hospital and the Methodology to Evaluate its Effectiveness. *Journal of Biomedical Informatics (JBI)*, *44*(6), 563-570.

Leichtenstern, K. & André, E. (2010). MoPeDT: features and evaluation of a user-centred prototyping tool. In: *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS '10*. New York, NY, USA: ACM. [93–102].

Nielsen, J. (1993). Iterative User Interface Design, *IEEE Computer*, *26*(11), 32–41.

Padma, T. & Balasubramanie, P. (2009). Knowledge based decision support system to assist work-related risk analysis in musculoskeletal disorder, *Knowledge-Based Systems*, *22*(1), 72–78.

Schreiber, G., Akkermans, H., Anjewierden, A., deHoog, R. , Shadbolt, N., deVelde, W. V. & Wielinga, B. (2001). *Knowledge Engineering and Management - The CommonKADS Methodology 2nd ed.*, MIT Press.

Tomic, B., Jovanovic, J. & Devedzic, V. (2006). JavaDON: an open-source expert system shell, *Expert Systems with Applications 31*(3), 595–606.

Corresponding Author

Martina Freiberg, Dept. of Artificial Intelligence and Applied Informatics, University of Würzburg, Am Hubland, 97074 Würzburg, Germany.
Tel. +49 931 3180465, freiberg@informatik.uni-wuerzburg.de