Template-based Extensible Prototyping for Creativity- and Usability-Oriented Knowledge Systems Development

Martina Freiberg and **Frank Puppe** [1]

Abstract. In knowledge-based systems (KBS) development, there still is a lack of research regarding user interface (UI) design and (usability) evaluation. Thus, especially KBS UIs still often are developed in a rather ad hoc manner, lacking reusability of proven solutions and potentially valuable experimentation with design alternatives and their thorough evaluation. We propose the tailored KBS prototyping and engineering tool *ProKEt* for practically supporting *Template-based Extensible Prototyping*, a technique for more efficient, affordable, and UI design/usability evaluation oriented KBS development. Further, we report current projects where both the approach and the tool provided valuable support.

Keywords: Knowledge-based System, Knowledge System Engineering, Extensible Prototyping, UI Design, Usability Evaluation

1 Introduction

Knowledge-based systems (KBS) engineering still constitutes an effortful, expensive task in terms of development time and costs; also, the focus often is on knowledge base development whereas UI design, creativity/experimentation, or even formal usability evaluation are considered rather lower priority task—if considered at all. Probably due to the numerous benefits of web-based systems, an increasing number of knowledge-based/expert systems seems to be developed for the web. However, such systems apparently often are being developed for quite specialized contexts in a rather ad hoc manner and not (re)using (neither providing) any patterns or best practices especially regarding the UI/interaction design. Amongst the reasons for this may be the lack of research—c.f. Duan et al. [9]—and tool support for encompassing KBS development, i.e., particularly integrating UI design and usability evaluation. An important premise for creative KBS (UI) development, for reusability of existing solutions and their usability-related evaluation is the availability of an affordable development methodology and tool. With regards to general KBS development there exist various software tools—such as JavaDON [15], or KnowWE [6]—as well as development methodologies—e.g., CommonKADS [14], or the Agile Process Model [5]. Yet, such approaches mostly focus on knowledge base design and evaluation. In contrast, we propose *ProKEt* as tailored development tool for web-based KBS that seamlessly couples agile KBS development—with particular focus on UI/interaction design—with semi-automated usability evaluation activities; therefore, the tool particularly supports *Template-based Extensible Prototyping*—a tailored form of evolutionary prototyping—and fosters reuse of existing KBS solutions. Concerning usability evaluation—specifically collecting click log data—there exist a vast range of both research-based and commercial tools; however, those mostly need to be separately installed and configured. In contrast, ProKEt seamlessly integrates appropriately tailored evaluation mechanisms.

In Section 2, we propose *Template-based Extensible Prototyping* in more detail. We then introduce the KBS engineering tool *ProKEt* in Section 3 for practical support of the described, tailored prototyping approach for KBS. In Section 4, we report experiences with the approach and the tool during current projects. We conclude with a summary of the presented research and an outlook on prospective future work in Section 5.

[1] University of Würzburg, Germany, email: freiberg/puppe@informatik.uni-wuerzburg.de

2 Template-based Extensible Prototyping

Evolutionary prototyping—see e.g. [7]—in particular evolves mature prototypes continuously into productive systems; yet the process, until a productive stage is reached may be quite lengthy.

Template-based Extensible Prototyping (TEP) We propose Template-based Extensible Prototyping (TEP) as a tailored form of online evolutionary [7] prototyping that additionally (re-)uses certain template or pattern sets for accelerating development. In contrast to basic evolutionary prototyping, TEP particularly focusses on the anytime production of functional systems. TEP basically consists of the two stages *pure prototyping*, and *productive prototyping*; consequently, it results in two types of prototype artifacts: An interactive, potentially slightly stripped-down user interface prototype (pure prototype), that can be transferred into an entirely productive, non-prototypical system with no effort. In the context of KBS, we think of pure prototypes as a specific excerpt of the system that mirrors only the core KBS specific UI and interactions, but not yet contains general required functionality such as session persistence or login mechanisms. In the productive prototyping stage, the pure prototype is transferred into a productive system by associating it with the respective knowledge base and aforementioned add-on functionality. For a detailed introduction of basic Extensible Prototyping and how it can be integrated with agile KBS development, see [12]. The additional usage of proven KBS solutions in the form of templates further enriches Extensible Prototyping by fostering efficiency and affordability as copying & and adaption/extension can be exploited. Templates thereby are applied directly at the pure prototyping stage when developing the UI of the prototype and future system, respectively. The range of templates should encompass more generic, system-level templates—e.g. for the entire framing UI design—to fine granular templates—e.g. for single UI elements such as buttons or the representation of questions and their answer alternatives. We propose a set of (system-level) templates derived

from practical project experiences in the next section. Besides from UI templates, knowledge patterns—for creating the knowledge base, such as proposed in [13]—are an opportunity for further leveraging the overall KBS development process.

Due to the application of reusable UI/KBS templates where reasonable and due to the deliberate exclusion of certain system aspects, pure prototyping becomes an affordable and straightforward task—even the more when TEP-tailored tools such as *ProKEt*, see Section 3, are available. Thus, it particularly supports the development of multiple alternative KBS prototypes in parallel and/or to develop in a highly iterative manner. Also, a more creative, experimental KBS design process is fostered, as e.g. novel KBS UI forms can be experimentally tried out while there is no need to deal with selecting—or newly developing—the appropriate required knowledge representation immediately. It can be argued, that template-based development and using a specific and thus potentially restricted tool could rather hinder than unfold creativity; there, we argue that it is no strict prerequisite to always make use of all or even any existing templates, but they are more to be seen as additional option to accelerate development in cases where system requirements and framing conditions are similar. Moreover, we claim it a major important feature of such template sets to be assembled of modular entities that built on each other and can be most easily extended; this allows for reusing just the templates that match the given requirements (and save time and efforts) and to get creative with other parts. Regarding template selection, this is currently intended as a manual process, depending on the project requirements and on the experiences of the knowledge engineer; however, we also plan to further enrich the approach with a template selection KBS which could—based on some entered framing properties—propose and setup the most appropriate template for a given context. Further, the affordability of frequent iterations supports usability-oriented development both implicitly and explicitly. Implicitly, as iterative development most often naturally detects shortcomings and flaws of the system which are more likely to be refined the more development iterations are performed. Explicit usability support is provided, as it becomes possible to create several alternative pure prototypes—which, as described above, exhibit a mature UI and the core interaction—and to formally evaluate them in a straightforward manner under quite realistic framework conditions. Due to the possibility to create alternative prototypes by simply adding adapted/other knowledge bases to the pure prototype, both UI and knowledge base can be assessed and refined in a highly iterative and visual manner.

Exemplary KBS UI Templates Due to practical experiences in past and current KBS projects, several system-level templates for web-based KBS could be identified. The **Questionary** style displays questions in resemblance to paper-based questionnaires. Two exemplary realizations of the Questionary template are shown in Figure 1, A (1-column style) & B (3-column style). For a more compact UI, the **Daily** template was developed; an exemplary 3-column Daily prototype is depicted in Figure 1, C. There, questionnaires and their included questions form a column-wide, visual entity similar as in common newspapers. Both Questionary and Daily style can be applied for documentation KBS—where the focus is on collecting data uniformly and correctly—as well as for consultation KBS—that derive one or several solutions based on the user input provided for the questions. Questionary and Daily style are introduced somewhat more elaborately in [12]. As an example for an efficient, skill-building KBS UI, we propose the **iTree** template, particularly apt for clarification consultation KBS—i.e., systems, where only a single issue is rated. An exemplary implementation is shown in Figure 2, A. The core issue as well as the questions–a tailored form of yes/no questions with additional value neutral/uncertain—that determine the core issue rating are presented in a hierarchical, tree-like manner. The core issue rating is derived from its top-level questions—placed directly underneath the core issue and are interactively and recursively navigable. We refer to [10] for a more detailed introduction of the iTree. Also applicable for clarification KBS, yet also for multiplex consultation KBS—where one issue/solution out of a potentially extensive set of solutions is to be derived due to the provided input—is the **One-Question** template. An example is shown in Figure 2, B. It basically aims at closely imitating a conversation between the system and a user by always presenting only the one appropriate next question at a time. The intention of such a strict conversational style is to ease the interaction as that way the user can always fully concentrate on the current question at hand, letting the KBS guide the problem solving workflow. In [10] also more details on the One-Question style are given. Of the proposed templates, so far only iTree and One-Question contain explanation modules, i.e., parts of the UI where the results of the KBS session are displayed and explained—in iTree above the tree part and in One-Question above the main, conversational question display panel. This is mostly due to the fact, that Questionary and Daily style were so far only applied in the context of documentation KBS where no solutions/diagnoses/explanations are required; nevertheless, there exist rough, alternative Questionary prototypes that also include prototypical explanation modules realized, e.g., as additional side panels.

3 ProKEt: Practical KBS Development Support

ProKEt is a tailored **Pro**totyping and **K**nowledge systems **E**ngineering **t**ool for web-based documentation and consultation KBS; it additionally provides support for various usability evaluation activities and fosters Template-based Extensible Prototyping (TEP). Pure prototypes are constructed in ProKEt by simply specifying a certain template name—e.g. *oqd* for the One-Question template—when defining the prototype-knowledge in a tailored XML format; then ProKEt automatically selects the required system-level and sub-templates and assembles them into a KBS prototype (pure prototyping). Templates thereby are defined by using the StringTemplate [4] technique, whereas the specific design/styling of UI elements is mostly done by separate CSS; relevant core interactivity—e.g., value abstraction—which needs to be imitated in pure prototypes is realized by JavaScript and is included automatically. When switching to productive prototyping, the basic KBS framework remains the same, making productive prototyping as easy as linking a productive knowledge base and potentially slightly adapting the base specification regarding, e.g., the CSS to be used. ProKEt currently supports exclusively *d3web* [1] knowledge bases which allow for defining a vast range of knowledge representations, such as (heuristic) rules, decision trees, or set-covering knowledge. This straightforward pure-to-productive-prototyping switch is supported for a bunch of basic KBS templates—as summarized in the previous section—out of the box. Thus ProKEt allows for a straightforward and affordable prototyping and engineering process in cases where framing conditions and system requirements are similar. Yet, also creativity is fostered, as existing templates and/or style files can easily be adapted or even completely rewritten, whereas the ProKEt framework—that finally assembles prototypes and productive KBS and enriches them by the required interactivity—needs not to be altered normally. For a more

extensive introduction of particularly the agile prototyping and engineering process with ProKEt and a detailed description of the tool, see [12]. It has to be noted, that when used as a prototyping environment alone, ProKEt (is not intended to and) does not provide any way to create (*d3web*) knowledge bases. However, when additionally using the semantic wiki KnowWE [6] for knowledge base development, both UI front-end and KB back-end can be developed in a tightly interconnected manner: Changes made to the knowledge base in the wiki can directly be deployed to the ProKEt artifact by a simple button click, making the changes immediately visible in the UI, which in turn eases the direct investigation of the recent changes and the potentially resulting side-effects regarding the UI.

Regarding usability, ProKEt directly offers integrated functionality to perform usability evaluations. This fosters the seamless integration of more or less extensive or formal evaluations into the KBS development process. Therefore, ProKEt basically offers *quantitative and qualitative* data collection mechanisms, which can be added for both prototypes and productive KBS by a simple property in the knowledge specification. As a result, e.g. questionnaires are included within the prototype UI and/or click logging is activated. Thus, developers can setup and conduct various evaluation scenarios and assess the current development state in a favorable way any time. Regarding *quantitative data*, ProKEt provides a tailored, mouse click and keyboard event logging mechanism that records all relevant actions during KBS usage sessions. Based on that data, ProKEt furthermore automatically calculates a bunch of known usability metrics—such as *Success Rate*, or *Average Task Time*. For *qualitative data* collection, ProKEt supports both the integration of form-based questionnaires/surveys—standard measures as e.g. the SUS [8] are provided out of the box, yet own questionnaires can be integrated equally easily—and of anytime feedback—a mechanism for collecting free user feedback at any time during a KBS session. All recorded data—quantitative as well qualitative—can be exported to a standard CSV format for further processing e.g. in statistical software. For more details on ProKEt's usability extension, see [11].

4 Case Studies

Several current projects so far showed the general applicability as well as the value of the Template-based Extensible Prototyping approach and the ProKEt tool.

Mediastinitis The Mediastinitis Registry [3] is a german national project for improving patient care in a cardiac medical context. Therefore, certain medical data are collected and statistically evaluated as to develop appropriate future treatment strategies—for more details, see [12]. For best supporting data entry by the medical staff, a *knowledge-based documentation system* was implemented. Based on a first specification of the underlying knowledge, the first prototype—Figure 1, A—was created; based on that, ProKEt allowed for creating also the two alternative designs in a straightforward manner by just adapting the respective UI templates and style files, and linking them with the existing knowledge. Thus the entire KBS framework, that was working for the first prototype, was reused, which greatly shortened the development efforts required for the UI alternatives (shown in Figure 1, B & C). After selecting the prototype fitting the requirements and expectations of the medical doctors best—Figure 1, B—a productive knowledge base was created and included with the chosen prototype UI (productive prototyping stage). In the further course, one of the doctors from the project reviewed the respectively current prototype by entering exemplary cases; the required adaptions—both regarding the knowledge base and its representation in the UI—were made in a timely manner and the expert continued reviewing the adapted prototype; thereby, the possibility to adapt UI and knowledge base separately from each other, but immediately re-merge them into new productive KBS for further reviewing was particularly valuable. This highly iterative process allowed for detecting and removing several non-obvious flaws regarding both knowledge base and UI, and thus for improving the system's overall usability.

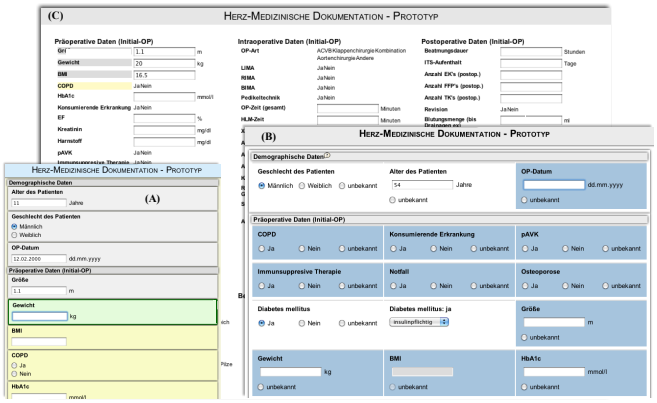

Figure 1. The three initial Mediastinitis prototypes (in german). 1-column questionary style (A); 3-column questionary style (B); daily style (C).

EuraHS EuraHS [2] is a project of the European Hernia Society (EHS) with the goal to improve patient care and increase knowledge regarding abdominal wall hernia surgery. Similar as in Mediastinitis, relevant data is to be collected and statistically evaluated; due to the similar basic framing conditions and application context, the first EuraHS prototype could be quickly built by (re)using the basic Mediastinitis prototype framework and just adapting the initial, exemplary knowledge specification. Based on that prototype, a first phase of iterative development began, where the expert participation remained passive, as he reviewed the respective prototypes and just reported what to refine. However, once the knowledge was transferred into a productive d3web knowledge base—starting the productive prototyping process—the expert was enabled to actively participate in the further development. This was possible due to the mechanism to immediately deploy adapted knowledge to the dialog system via the direct linkage between the knowledge base development tool KnowWE [6] and the dialog UI. This extensive expert participation was perceived highly beneficial as it led to a high satisfaction on the side of the expert due to his active involvement and resulting identification with the system; it further saved time and efforts, as on the one hand the expert knowledge was formalized in an unsophisticated manner and thus contained less flaws, and on the other hand the parallel development of KBS/UI (university team) and KB (expert) led to quicker overall results. The highly iterative process again enabled many KB and UI refinement cycles, thus enhancing the overall quality of the system. The final EuraHS implementation is quite similar to the final Mediastinitis system—c.f. Figure 1, B—however enhanced by several additional features including image questions (where answers can be selected visually) and a more comprehensive mechanism for flexibly fading in and out parts of the UI depending on already provided answers. For a more detailed description of EuraHS, see [12].

JuriSearch *JuriSearch* was started in 2012 as a cooperation between the university of Würzburg and the RenoStar corporation and aims at building a freely accessible, web-based knowledge-based system for the legal domain for various topics, such as right of cancellation or the law of tenantry. The target system is intended to integrate both a standard consultation (entrance) module—helping the user to preselect the specific problem definition—and various clarification modules for each potential problem—which then validate the concrete rating of that issue. Target users range from legal laymen—searching for a basic understanding/estimation of their case to (fresh) lawyers seeking for guidance regarding legal (sub)domain(s) that are not exactly their special field of work. So far, the focus lay on

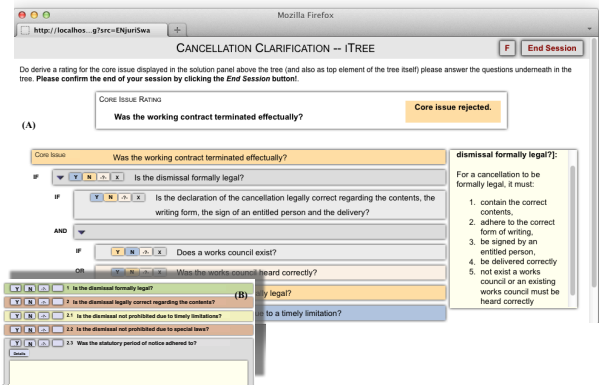

Figure 2. The two JuriSearch prototypes: interactively navigable iTree clarification style (A), and One-Question clarification style (B).

the clarification modules each of which rates exactly one distinct core issue, e.g. "Was the cancellation legally correct" (labour legislation domain). Initially, we experimented with two alternative yet distinct UI forms: An iTree implementation, depicted in Figure 2, A, and a One-Question UI, depicted in Figure 2, B. Therefore, first an *iTree* prototype was implemented based on a rough specification of the underlying knowledge. The possibility, to create various prototypes by simply exchanging the knowledge specification again proved valuable, as that way the prototypes could be reviewed highly iterative by a RenoStar staff member; this strongly supported the refinement and correction of both the underlying knowledge but also its most appropriate UI representation. ProKEt further allowed for creating the alternative One-Question UI in an affordable and timely manner in parallel to the iTree development. Based on those two alternative prototypes, so far several comparative assessments were performed. As first goal of the studies, it was assessed whether the iTree or the One-Question UI style were more suitable—if any—for the target context in general; there, the results of the studies indicate, that for the specific context of legal clarification consultation—a domain of highest expertise which needs to be mirrored adequately yet understandably by the KBS—the iTree is perceived more suitable and intuitively usable than the One-Question UI. Elaborate details on that study can be found in [11]. Furthermore, studies were conducted as to asses two distinct alternative knowledge base structures for the iTree style—one adhering to a legal specialist deduction scheme, the other specifically intended to provide more guidance and overview for legal laymen users; there, so far no significant distinction could be identified whether one scheme works better than the other. However, both the knowledge base as well as the UI could be drastically improved by refining them according to the respective insights and user comments gained in the user studies.

5 Conclusion

For leveraging the issue of a lacking integration of UI-related creativity and usability activities in KBS current development, we proposed *Template-based Extensible Prototyping* as KBS development technique that despite originally being developed specifically for the KBS domain may as well be applicable in general software engineering. For practical support of the approach, we introduced the KBS engineering tool ProKEt and we reported case studies that showed the applicability and value of the approach and tool. Regarding future work, current and upcoming projects raised the need for extending the collection of KBS classes and UI templates supported by ProKEt. Also, integrating mouse tracking mechanisms as addition to the existing click logging seems promising as to gain even more detailed insights regarding the UI usage evaluation. Equally, an automated, visual evaluation aid—that compares the solutions derived by the users with the correct solutions—could strongly support usability related evaluations. Further, a more formal classification of existing KBS types and respective suitable UI styles/interaction forms—e.g. in the form of a KBS pattern catalogue or also an interactive pattern selection KBS—could enrich the overall approach; thereby, the combination of UI templates/patterns and KB patterns [13] seems promising for encompassing, reusability-enabling KBS development.

REFERENCES

[1] http://d3web.sourceforge.net/ , last checked Jun. 1st, 2012.
[2] http://eurahs.drwontwikkeling.nl/, last checked Jun. 1st, 2012.
[3] http://www.dgthg.de/register, last checked Jun. 1st, 2012.
[4] http://www.stringtemplate.org/, last checked Jun. 1st, 2012.
[5] Joachim Baumeister, *Agile Development of Diagnostic Knowledge Systems*, IOS Press, AKA, DISKI 284, 2004.
[6] Joachim Baumeister, Jochen Reutelshoefer, and Frank Puppe, 'KnowWE: A Semantic Wiki for Knowledge Engineering', *Applied Intelligence*, **35**(3), 323–344, (2011).
[7] Michel Beaudouin-Lafon and Wendy Mackay, 'Prototyping Tools and Techniques', in *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, pp. 1006–1031, Hillsdale, NJ, USA, (2003). L. Erlbaum Associates Inc.
[8] J. Brooke, 'SUS: A quick and dirty usability scale', in *Usability evaluation in industry*, eds., P. W. Jordan, B. Weerdmeester, A. Thomas, and I. L. Mclelland, Taylor and Francis, London, (1996).
[9] Y. Duan, J. S. Edwards, and M. X. Xu, 'Web-based expert systems: benefits and challenges', *Information & Management*, **42**, 799–811, (September 2005).
[10] Martina Freiberg and Frank Puppe, 'itree: Skill-building user-centered clarification consultation interfaces (to appear)', in *KEOD 2012 - Proceedings of the International Conference on Knowledge Engineering and Ontology Development*, (2012).
[11] Martina Freiberg and Frank Puppe, 'Prototyping-based Usability-oriented Knowledge Systems Engineering', in *To appear in Proceedings of Mensch und Computer 2012*, (2012).
[12] Martina Freiberg, Albrecht Striffler, and Frank Puppe, 'Extensible prototyping for pragmatic engineering of knowledge-based systems', *Expert Systems with Applications*, **39**(11), 10177 – 10190, (2012).
[13] Frank Puppe, 'Knowledge Formalization Patterns', in *Proceedings of PKAW 2000, Sydney Australia*, (2000).
[14] Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Robert de Hoog, Nigel Shadbolt, Walter Van de Velde, and Bob Wielinga, *Knowledge Engineering and Management - The CommonKADS Methodology*, MIT Press, 2 edn., 2001.
[15] Bojan Tomic, Jelena Jovanovic, and Vladan Devedzic, 'JavaDON: an open-source expert system shell', *Expert Systems with Applications*, **31**(3), 595 – 606, (2006).