Interaction Pattern Categories

Pragmatic Engineering of Knowledge-Based Systems

Martina Freiberg, Joachim Baumeister, and Frank Puppe

University of Würzburg, Institute of Computer Science
Dept. of Artificial Intelligence and Applied Informatics
D-97074 Würzburg, Germany
freiberg/baumeister/puppe@informatik.uni-wuerzburg.de

Abstract. The application of knowledge-based consultation- and documentation systems is, apart from large industrial projects, often also beneficial for small to mid-sized enterprises. Yet, their design and implementation still is a tedious and costly task. We motivate, that customized UI and interaction patterns constitute a pragmatic technique for supporting especially requirements engineering, and thus are capable of considerably promoting real-world projects. In this paper, we introduce abstract categories—*Guided-*, *Adaptive-*, and *Autonomous Entry*—for classifying tailored patterns for knowledge-based systems. Further, we discuss their role in an overall approach extending the *Agile Process Model* and resulting benefits.

Keywords: Dialog System, User Interface Design, Agile Development

1 Introduction

Knowledge-based systems gained increasing impact also outside academia over the last decades. Apart from large clinical and industrial projects, the application of knowledge-based consultation and documentation systems is also often beneficial for small to mid-sized corporations. Yet, the trade-off between their potential benefits and their mostly still tedious and costly development, is still often perceived as unfavorable, and respective projects are declined.

In general software engineering, *user interface (UI) prototyping* already is an established methodology regarding iterative, rather inexpensive system specification before the final product is implemented [3]. Also, UI prototyping permits the early evaluation of (several) design options. Inspired by that approach, we suggest *Interaction Patterns for Knowledge-Based Systems* as a cornerstone of a tailored, agile knowledge system development methodology. The overall approach integrates pattern- and prototyping-based development into an existing, agile process model, and thus combines the advantages of reusing approved solutions (patterns) and of affordable, iterative system specification (UI prototyping within an agile process model). We argue, that this constitutes a rather pragmatic way to enhance understanding, discussing, and specifying system requirements at project start. This in turn helps to promote respective projects in the first place, and thus renders its application especially interesting when addressing small to mid-sized enterprises as customers.

As a first step into this direction, this paper introduces *Interaction Pattern Categories*, that provide an abstract classification framework for knowledge systems and corresponding interaction/UI patterns. We further discuss the role of such patterns within the proposed, extended agile approach; the details regarding the prototyping and a respective tool are subject of separate work, see [7].

The rest of the paper is organized as follows: Related research is presented in Section 2. In Section 3, we introduce our classification framework of *Interaction Pattern Categories* and the relevant terminology. We further present three categories, identified on the basis of past experiences with conducted projects. In Section 4, we outline the extended, agile process model, and the patterns' specific role as well as resulting benefits. We conclude with a summary of the presented work and a discussion of future research directions in Section 5.

2 Related Work

The process model for knowledge system development, that we suggest in this paper, integrates pattern- and prototyping-based development, thus uniting the advantages of both approaches and especially fostering an enhanced requirements engineering.

Patterns specify proven solutions for recurring (design) problems and are established in many domains: Examples are *software engineering*, *ontology engineering*, or *knowledge formalization*, [8, 9, 14]. They offer the advantage to reuse approved solutions for similar problems, and thus to reduce development efforts and to profit from the lessons learned. Regarding *UI–/interaction design*, tailored, domain-specific pattern collections exist, e.g., [16–18]. Yet, patterns originating from such research cannot be straightly transferred to our context, as knowledge-based systems put specific demands on interaction and UI design.

Regarding knowledge system development, various methodologies have been proposed in the past—see [12] for an overview. More recent works emphasize the relevance of *agility*, e.g., see [2, 10]. We follow that direction by integrating pattern- and prototyping-based techniques into an agile process model. Previous approaches, however, often strongly emphasize the development of the knowledge itself. In contrast, we specifically support knowledge system UI and interaction design by the means of tailored patterns and prototyping as to enhance requirements engineering on the one hand, and to foster a pragmatic, affordable promotion and execution of respective projects on the other hand.

UI prototyping so far has become an established approach in general software engineering [3] as well as in HCI and usability engineering [4]. Main advantages are a strong support of requirements specification, and the opportunity to evaluate (several) UI design(s) at an early stage. In [11], a prototyping tool, that incorporates design patterns for layout support, has been proposed. Though generally related to our approach, that work focusses on cross-device design of general web-style interaction. Contrastingly, we explicitly consider UI and interaction design of knowledge-based consultation and documentation systems.

3 Interaction Pattern Categories

By *Interaction Patterns for Knowledge-Based Systems*, we understand the description of the systems' interaction- and UI design for a specified context. They comprise a compact specification of their applicability, and exemplify the corresponding solution approach. Yet, there exist attributes, the value of which may be common to more than one distinct pattern. Thus, we first introduce an abstract framework—*Interaction Pattern Categories*—for classifying patterns according to such common properties before specifying concrete patterns. In Section 3.1, we first introduce relevant terminology, and in Section 3.2, we present the classifying categories—*Guided-*, *Adaptive-*, and *Autonomous Entry.*

3.1 Relevant Terminology for Specifying Pattern Categories

In the following, we specify the addressed system types as well as the classifying attributes, that characterize the pattern categories, in more detail.

Knowledge-Based Systems: We specifically address *knowledge-based systems* with our approach—by that, we understand knowledge systems, that serve either a *consultation* or a *documentation* task. In both cases, the main user-system interaction is structured data entry—mirrored by *"Entry"* in the pattern category names. Regarding consultation, the system gradually derives solutions for a given problem with the respective, implemented reasoning mechanisms based on the provided user input (answers). Documentation systems emphasize supporting uniform and reliable data input as effectively as possible.

Classifying Attributes: The attributes *User Competence*, *Context Presentation*, and *Data Volume* are common to all patterns of one category. Major classifier thereby is *User Competence*—in the context of knowledge-based systems, lengthy, strictly prescribed interviews can be annoying and inflexible for competent users, that might want to influence the interrogation flow according to their expertise. This makes it essential to tailor the system and interface design to the target users' competence.

A. User Competence: A *naive* data provider follows the prescribed interrogation sequence, with no desire for deviation or adaption; possible reasons can vary from rather low domain competence/lacking experience to a highly stressful usage context (but nevertheless domain expertise). *Experienced* users possess a certain domain expertise, and thus may be interested in system-suggested workflow guidance, yet, additionally require the option to influence the interrogation and to deviate from suggested paths. An *autonomous* problem solver finally possesses sufficient expertise to solve the problem independent from system guidance, based on the (potentially various and complex) information presented.

B. Data Volume: The amount of data that is processed during a typical interrogation session; thus, it corresponds to the number and the complexity of questions required for deriving a solution or entering a complete data set. We roughly distinguish between *small*, *medium*, and *large*. *Data Space*, in contrast, specifies the universal range of possible input data, and thus corresponds to the domain complexity. The respective data volume/data space combination does not influence the pattern categorization, yet the knowledge required for a specific implementation—e.g., large data space and low data volume implies sophisticated interrogation structures to present appropriate questions efficiently.

C. Context Presentation: *No context* means, that during an interrogation, only the required questions are presented, but no further information. Otherwise, we distinguish *support knowledge*—auxiliary information (not interrogation specific), or informal knowledge representations—and *interrogation context*—i.e., additional information regarding, e.g., the progression of the workflow, or indicating the consequences of choosing certain answer alternatives in advance. Type and extent of context presentation highly depend on the respective level of user competence—concerning naive data providers, interrogation context often is not required, yet for complex questions, support knowledge presentation might be advisable; with rising competence, the presentation of interrogation context gains importance for supporting an independent, efficient system usage.

3.2 Interaction Pattern Categories for Knowledge-Based Systems

Based on experiences from past projects, we define three basic categories for UI/interaction patterns: *Guided-*, *Adaptive-*, and *Autonomous Entry*. For each category, we describe the *Problem Statement*, the *Solution*, and the *Applicability*, specifying common properties that apply to all contained patterns. Further, we provide *Examples*—i.e., existing implementations—and *Variants*, that describe in what regard patterns of a category may vary.

The basic interaction specifying each pattern, regards question selection during interrogation. Even if patterns later vary, e.g., regarding the processed data volume, that basic interaction remains the same. For its specification we use the UML sequence diagram notation and the elements: *User*, the system *Interface*, *Questions* (presented to and answered by the user), the *Data* pool (storing data resulting from provided answers or reasoning), and the *Knowledge* component.

A. Guided Entry

Problem Users act as naive data providers, thus for a reliable, effective decision- or documentation support, a high level of system autonomy/guidance regarding the interrogation flow is required. Data volume might vary from small to medium.

Solution An interview metaphor is transferred to the interface, where the user and the system interact alternately. The system flexibly reacts to the provided answers by adapting the interrogation sequence, thus presenting only the question(s) that fit the respective context best. The interview proceeds system-guided, and deviation is mostly not (or only in limited terms) intended. Thus, presenting interrogation context is not mandatory, even though regarding lengthy sequences status feedback may be beneficial. Contrastingly, support knowledge is required in the case of complex/difficult questions for clarification. Figure 1 depicts the interaction sequence for *Guided Entry.* The interface initi-

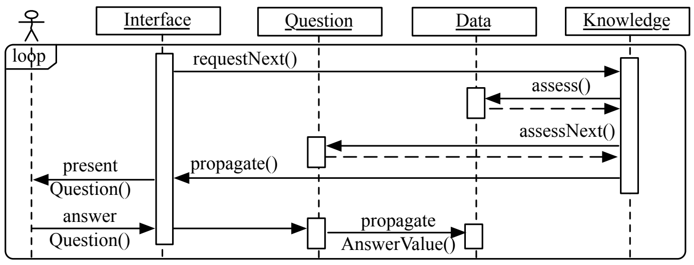


Fig. 1. Guided Entry—Basic interaction sequence for question selection.

ates the question request, whereupon the knowledge component assesses the next question—where available, based on the previously provided user input stored in the data pool—and propagates the result back to the interface. The then provided answer of the user is propagated to the data component and thus made available for the knowledge component hereafter. Those steps are performed iteratively until a defined interrogation sequence is finished.

Applicability Systems based on *Guided Entry* equally fit consultation and documentation tasks. Especially documentation of high quality or frequently recurring data is supported, as specified data entry can be assured by the strict, system-guided interrogation flow. However, if a higher level of user autonomy is desired—e.g., influence on the interrogation, or adaptable question representation—*Adaptive-* or *Autonomous Entry* provide more flexibility.

Examples Figure 2 presents two implementation variants of *Guided Entry. CareMate* (A) is a quick response second-opinion system for emergency situations. Its one-question interaction style creates the literal impression of an interview and supports the intuitive usage in the context of stressful emergency conditions. Continuous status feedback on the current solution states is provided, and the processed data volume is rather small. For a more detailed introduction, see [1]. *SonoConsult* [15] is a consultation and documentation system for the field of abdominal ultrasound. The multiple-question interaction style resembles a paper-based interview (questionary) and helps to cater with the rather large data volume. Both support knowledge (question clarification) and interrogation context (presenting currently derived solutions) are provided.

Variants Pattern variants arise with regards to the type and extent of context

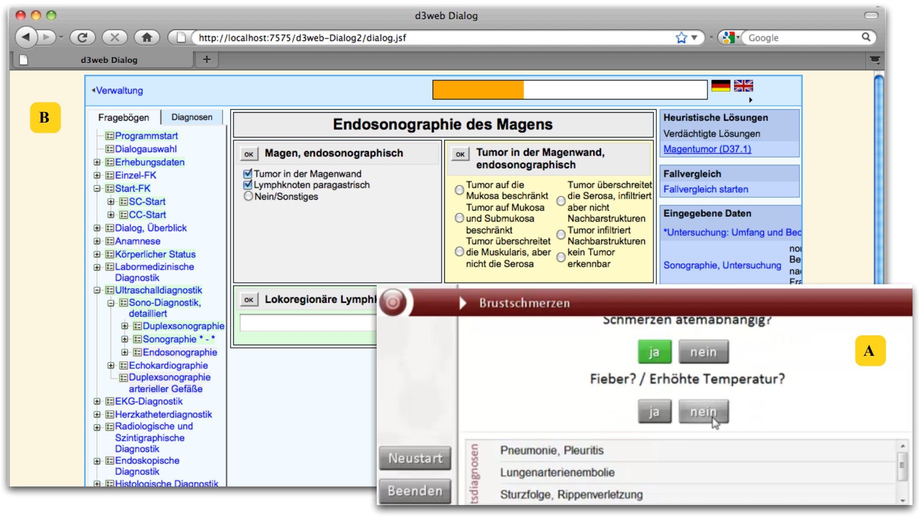


Fig. 2. Implementation variants of *Guided Entry*, both in german: A) Digitalys CareMate and B) SonoConsult.

presentation (see above examples), as well as regarding the characteristics of the *naive* user (e.g., expert in stressful context vs. non-expert's ad-hoc usage).

B. Adaptive Entry

Problem Experienced target users have a certain—yet, from user to user potentially varying—domain competence; consequently, both system guidance as well as the option for autonomous decisions regarding the workflow are desired. Also, questions should be presented in a user-adaptable manner.

Solution The system basically suggests the most appropriate workflow to the user; yet, also the option to deviate from that path and choose an adapted interrogation sequence is provided. Where applicable, a hierarchical tree metaphor is applied to cater with varying user competence levels: Questions are defined both on an abstract (aggregate) level, but also subdivided into (several) refined questions, where reasonable. Thus, according to their expertise, users may either answer the aggregate questions—taking less time, but requiring more expertise—or request the presentation of the questions' refinement. To support the user's decision-making, providing interrogation context is strongly recommended. Also, depending on the refinement level and complexity of the questions, support knowledge should be additionally presented.

Figure 3 sketches question selection in *Adaptive Entry*. Basically, the user decides whether to follow the system-guided interrogation or whether to choose an own path. Regarding the first alternative, question selection proceeds as in *Guided Entry* (Figure 1). In the second case, either the user's competence allows for an-

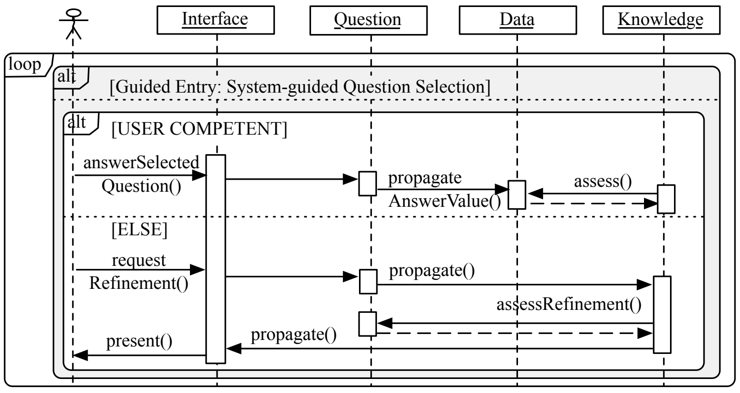


Fig. 3. Adaptive Entry—Basic interaction sequence for question selection.

swering the currently displayed question; then the answer is propagated to the data pool and thus is available for the knowledge component as the interrogation continues. Otherwise, the user can request question refinement whereupon the knowledge component assesses the possible refinement, and propagates the result back to the interface for displaying it to the user.

Applicability Apart from consultation, respective systems can, with limitations, also serve documentation purposes. In that case, special care has to be taken that all required input data is obtained from the user. Regarding effective interrogation of naive data providers *Adaptive Entry* is only marginally suitable.

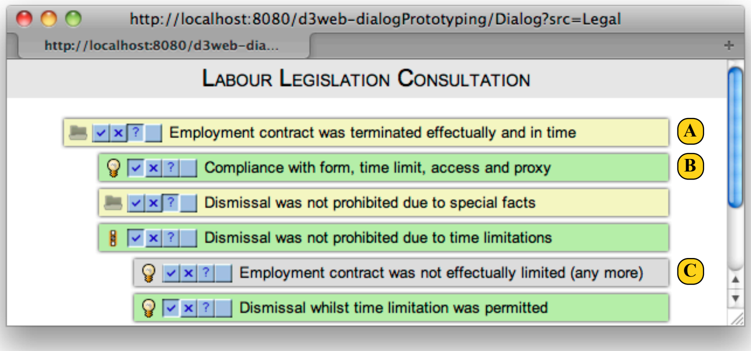
Examples Figure 4 shows the Labour Legislation Consultation, that clarifies, whether a dismissal in a given context is legitimate. Figure 4, A, represents the problem statement. Its current derivation state and the questions' state (e.g., answered) are visually indicated by background coloring and updated with each provided answer. Questions can be processed either in the sequence suggested (i.e., from top to bottom), or in any other order. Further, adaptable question presentation is implemented—e.g., Figure 4, B, was confirmed on the abstract level; question *Dismissal was...* is expanded into refined questions (Figure 4, C).

Variants Possible variants originate from different forms of context presentation as well as from different data volumes that may be processed.

C. Autonomous Entry

Problem Target users are highly competent, autonomous problem solvers, thus no explicit guidance regarding the interrogation sequence is required.

Solution The user explores the (various and potentially complex) information sources presented by the system. Integrated knowledge-based components—e.g., consultation features or automated data entry support—can be used optionally, but are not mandatory to benefit from system usage. The user provides any in-


http://localhost:8080/d3web-dialogPrototyping/Dialog?src=Legal

http://localhost:8080/d3web-dia...

LABOUR LEGISLATION CONSULTATION

- Employment contract was terminated effectually and in time (A)
  - Compliance with form, time limit, access and proxy (B)
  - Dismissal was not prohibited due to special facts
  - Dismissal was not prohibited due to time limitations
    - Employment contract was not effectually limited (any more) (C)
    - Dismissal whilst time limitation was permitted

Fig. 4. Labour Legislation Consultation—Indication of the solution and its current rating (A), clarifying questions (B), and further refinement of one question (C).

put data voluntarily; based on those data fragments, the system performs rather modularized reasoning, following sort of a *bits and pieces* metaphor. Thus, the system merely provides a second-opinion to the user in presenting reasoning results (e.g., rated solutions, next-input suggestions). Such extensive user control requires a high level of context presentation, regarding both types of context.

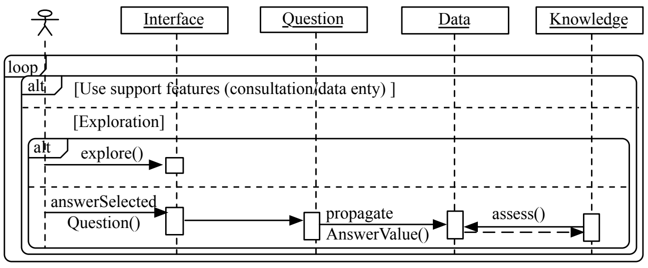


Fig. 5. Autonomous Entry—Basic interaction sequence for question selection.

As Figure 5 shows, the user always can choose between using more formal knowledge components and free exploration. In the first case, potentially any kind of (complex) knowledge component can be integrated into such a system, e.g., according to the *Guided Entry* or *Adaptive Entry* categories. Otherwise, the user can either simply explore the provided information, or answer questions autonomously in a modularized manner. Answers then are propagated to the data component, and from there assessed by the knowledge component; the latter rates solutions, presents context, and recommends next steps piecemeal.

Applicability *Autonomous Entry* can be applied for loose consultation, as well as regarding a more informal, potentially collaborative documentation task. In contrast, it is inappropriate, if rather naive data entry is desired, as no strict workflow guidance for solving the addressed problem is provided. Further it is not suitable for high quality documentation tasks, as any interaction takes place voluntarily, and thus the supply of any data cannot be guaranteed.

Examples Implementation examples are the user-centered consultation approach described in [5], the PEN-Ivory system [13], but also the *Inline Answering* concept provided by the Semantic Wiki KnowWE [1].

Variants Implementation variants arise with respect to the data volume, and the type and extent of context presentation. Despite mainly addressing expert users, systems falling into this category, might to some extent also be suitable for unexperienced ad-hoc usage. Finally, resulting systems can vary regarding the extent of integrating knowledge-based features.

The proposed pattern categories classify basic knowledge system types and corresponding UI/interaction design patterns according to the level of user competence (corresponding to the level of system guidance). Ongoing research addresses the definition of concrete patterns and their categorization accordingly. We proceed by discussing how such patterns can be integrated in an extended, agile process model for developing knowledge-based consultation and documentation systems.

4 Pragmatic Knowledge System Engineering

Regarding knowledge system development and knowledge engineering, there exist diverse approaches today, such as *CommonKADS*, *MIKE*, or adaptions of the classical *stage-based* and *incremental* software development models. Yet, for the success of knowledge system projects in the context of small to mid-sized companies, a pragmatic approach—affordable and efficient regarding time and effort—is essential, c.f. [2]. Especially for promoting such projects in the first place, it is important to quickly come up with first solutions, e.g., in the form of prototypes or example implementations. In this respect, we made positive experiences with applying the *Agile Process Model*, described in [2]. However, that model emphasizes knowledge base development, not yet taking much into account the design of the target system's interface, or usability traits. In extending this model, not only UI/interaction design gains importance in the overall development process, but also the integration of usability activities.

In the following, we introduce the *Extended Agile Process Model*, and afterwards we discuss resulting benefits specifically regarding the integration of tailored UI/interaction patterns. Although prototyping and usability-related activities are included in the model for reasons of completeness, their detailed discussion is part of further work, see [7].

4.1 The Extended, Agile Knowledge System Engineering Model

Figure 6 outlines the *Extended Agile Process Model*—the gray parts represent the original model, consisting of the four phases *System Metaphor*, *Planning Game*, *Implementation*, and *Integration*. For a detailed discussion, see [2].

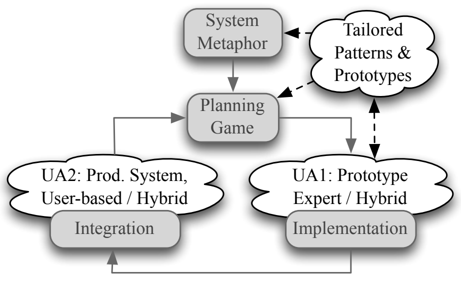

System Metaphor
Tailored Patterns & Prototypes
Planning Game
UA2: Prod. System, User-based / Hybrid
UA1: Prototype Expert / Hybrid
Integration
Implementation

Fig. 6. Extended Agile Process Model.

Basically, tailored patterns and prototyping can support both *System Metaphor* and *Planning Game*. In *System Metaphor*, the system objectives are defined by developers and customers. Based on appropriate patterns and corresponding implementation examples, a basic idea can be developed more easily. Thereby, patterns can be assessed either manually, or by using a tailored recommender system, that suggests patterns matching the target context. Prototypes, that also provide the relevant user-system interactions, further support that process by presenting a realistic simulation of a potentially resulting system as opposed to the static, visual depiction of knowledge system examples provided by the patterns. The *Planning Game* defines the scope and prioritization of development tasks. Here, patterns ease the analysis and valuation of system requirements—taking place during the *Exploration* sub-phase of the planning game—by providing clear specifications of required features and interactions. Additionally, prototyping supports that task by allowing for actually trying out (and thus better evaluating) relevant functionalities.

With regards to *Usability Activities*, the original model can be extended both regarding *Implementation* and *Integration* (Figure 6, UA1, UA2). The basic model defines *Implementation* as a test-first activity—i.e., before actually implementing new or additional features, the corresponding tests for assuring their correctness are developed. This can be expanded by an evaluation-first activity, in the sense that based on the formerly created prototypes, usability issues are assessed and valued first, before continuing with test-first implementation as defined by the model. Without going into detail here, at that stage, expert- or hybrid approaches (according to a categorization suggested in [6]) seem to be most appropriate. During *Integration*, the implemented functionality is added to the productive system, using integration tests for assuring its overall correctness and integrity. Such testing can be extended by usability checks that evaluate, whether the system still meets the specified usability goals. As this results in a running version of the productive system, not only hybrid, but also purely user-based usability evaluation can be beneficial.

4.2 Benefits

The integration of tailored patterns into an extended agile model offers several benefits: First, the patterns uniformly specify common framework conditions of different knowledge-based system types; thus, they provide a descriptive and visual language, that enables customers and developers to discuss at the same competence level. This fosters a clear communication and thus reduces potential misconceptions right away, that otherwise can lead to additional, unnecessary redesign cycles. Next, the patterns present actual implementation examples, that can be assessed, and serve as a inspirational source regarding the concrete project at hand. Even in case none of the provided patterns or examples completely satisfy the project- and customer requirements, those nonetheless are helpful by providing an overview of the possibilities and a basis for further discussions. Despite diverse general pattern collections and UI prototyping tools, to date to the best of our knowledge no tailored patterns/tools exist addressing specifically knowledge-based systems. As the latter exhibit quite specific characteristics, our approach can provide strong support for their development.

5 Conclusions and Future Work

We motivated that tailored patterns can constitute the cornerstone of an extended, agile model for knowledge system development. Especially when targeting smaller to mid-sized companies as customers, the suggested approach is a rather inexpensive, pragmatic technique for promoting and launching respective projects. As a first step, in this paper we introduced three abstract categories for classifying corresponding patterns. Due to our focus on knowledge-based consultation and documentation systems, those categories specifically address the data entry task; yet, the elementary classification—*guided, adaptive,* and *autonomous interaction* might be applied accordingly for other forms of interaction. The categorization arose from practical experiences with implementing knowledge-based systems in the past, such as *SonoConsult* [15], *Digitalys CareMate*, or more recently the *Semantic Wiki KnowWE* [1]. Further research includes the question, whether additional pattern categories are required. Based on those, as well as on an assessment of further existing systems, concrete patterns will be specified. Currently, also a tailored prototyping tool is developed [7], that will be further extended based on an analysis of required knowledge system base components.

References

1. Baumeister, J., Reutelshoefer, J., Puppe, F.: KnowWE: A Semantic Wiki for Knowledge Engineering. Applied Intelligence (2010)
2. Baumeister, J., Seipel, D., Puppe, F.: Agile development of rule systems. In: Giurca, Gasevic, Taveter (eds.) Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches. IGI Publishing (2009)

3. Bäumer, Dirk and Bischofberger, Walter R. and Lichter, Horst and Züllighoven, Heinz: User Interface Prototyping—Concepts, Tools, and Experience. In: ICSE '96 Proceedings of the 18th International Conference on Software Engineering. pp. 532–541 (1996)
4. Beaudouin-Lafon, M., Mackay, W.: Prototyping tools and techniques. In: The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications. pp. 1006–1031. L. Erlbaum Associates Inc., Hillsdale, NJ, USA (2003)
5. Buscher, G., Baumeister, J., Puppe, F., Seipel, D.: User-Centered Consultation by a Society of Agents. In: Proc. of the 3rd International Conference on Knowledge Capture (K-CAP 2005), Banff, Alberta, Canada (2005)
6. Freiberg, M., Baumeister, J.: A survey on usability evaluation techniques and an analysis of their actual application. Tech. Rep. 450, Institute of Computer Science, University of Würzburg, Germany (2008)
7. Freiberg, M., Mitlmeier, J., Baumeister, J., Puppe, F.: Knowledge system prototyping for usability engineering. In: Proceedings of the LWA-2010 (Special Track on Knowledge Management) (2010)
8. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley Longman (1995)
9. Gangemi, A., Presutti, V.: Ontology Design Patterns. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies (2009)
10. Knublauch, H.: Extreme programming of knowledge-based systems. In: Proceedings of eXtreme Programming and Agile Processes in Software Engineering (XP2002) (2002)
11. Lin, J., Landay, J.A.: Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. In: CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems. pp. 1313–1322 (2008)
12. Plant, R., Gamble, R.: Methodologies for the development of knowledge-based systems, 1982–2002. Knowledge Engineering Review 18(1), 47–81 (2003)
13. Poon, A.D., Fagan, L.M., Shortliffe, E.H.: The PEN-Ivory Project: Exploring User-Interface Design for the Selection of Items from Large Controlled Vocabularies of Medicine. Journal of the American Medical Informatics Association pp. 168–183 (1996)
14. Puppe, F.: Knowledge Formalization Patterns. In: Proceedings of PKAW 2000, Sydney Australia (2000)
15. Puppe, F., Atzmueller, M., Buscher, G., Huettig, M., Luehrs, H., Buscher, H.P.: Application and evaluation of a medical knowledge system in sonography (sonoconsult). In: Proceeding of the 2008 conference on ECAI 2008. pp. 683–687. IOS Press, Amsterdam, The Netherlands, The Netherlands (2008)
16. Ratzka, A.: Identifying user interface patterns from pertinent multimodal interaction use cases. In: Mensch & Computer 2008: Viel Mehr Interaktion. pp. 347–356 (2008)
17. Schmettow, M.: User interaction design patterns for information retrieval systems. In: EuroPLoP' 2006, Eleventh European Conference on Pattern Languages of Programs. pp. 489–512 (2006)
18. Tidwell, J.: Designing Interfaces — Patterns for Effective Interaction Design. O'Reilly Media Inc. (2006)