

The Usability Stack: Reconsidering Usability Criteria regarding Knowledge-Based Systems

Martina Freiberg, Joachim Baumeister, Frank Puppe

University of Würzburg

D-97074, Würzburg, Germany

freiberg/jobapuppe@informatik.uni-wuerzburg.de

Abstract

Considering usability-based design and evaluation, many general guidelines and heuristics have been proposed in the past. Yet there is still a lack of tailored criteria for the specific case of knowledge-based systems. In this paper we propose the *Usability Stack* for knowledge-based systems as a new model that summarizes and classifies usability criteria for that particular context. The model as a whole, as well as the different layers in particular, are discussed. We also shortly describe the results of exemplarily going through one knowledge system, developed by our department, according to the model.

1 Introduction

Concerning the usability, knowledge-based systems constitute a particular case compared to general software.

Concerning the case of consultation, knowledge-based systems often are not used by their potential users on a regular basis and additionally under considerable strain. This is confirmed by our experience with systems that are applied in the medical domain—examples are systems that support abdomen sonography (SonoConsult, [5]) or dental consultancy (a tailored dialog, used as a case study in [2]). Therefore it is essential that knowledge-based consultation systems are as easily usable and as highly self-descriptive as possible. That way, users can always resume working with the system in an easy and effective way—even after longer or stressful breaks.

Another issue, concerning especially the knowledge acquisition task, is the required high level of expertise concerning both the application domain and knowledge engineering in general. Thus, ideally, knowledge engineers and domain experts collaborate on setting up a knowledge-based system; yet, a cooperation of—a minimum of—two such specialists is often too expensive, for instance see Puppe [13]. In drastically simplifying the usage of knowledge-acquisition tools, also non-experts could more easily support domain specialists in developing knowledge-based systems.

Thus, enhancing the ease of use of knowledge-based systems—regarding both the consultation and knowledge-acquisition use case—could contribute to further extend the range of potential application contexts, and thus increase their distribution. Hence the important, still unmatched, long-term goal is to purposefully enhance their *usability*.

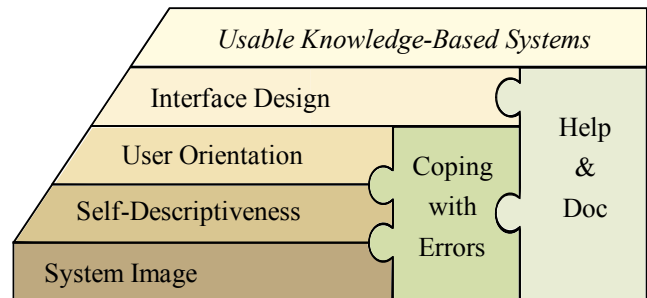


Figure 1: *Usability Stack* for knowledge-based systems

Regarding usability-based design and evaluation, many usability resources have been proposed to date. Yet, concerning knowledge-based systems, those resources often either are lacking specific requirements, or on the other hand partly incorporate insignificant aspects.

In this paper, we propose the *Usability Stack* for knowledge-based systems (depicted in Figure 1)—a model of usability criteria, tailored for knowledge-based systems. By knowledge-based systems we mean systems that utilize problem-solving knowledge and inference mechanisms to support the user in decision-making or in finding solutions for a given problem. At that point in time, we aim at generating a model that is applicable to different classes of knowledge-based systems with respect to their underlying inference method—that is, we made no distinction between, for example, case-based or rule-based knowledge systems so far. Whether and to what extent such a distinction could be additionally useful for our context will be further investigated in the future.

To create a basic stack-model of usability criteria, we reconsider well-known usability resources in detail; as far as necessary, we extend and reassemble usability criteria, which are characterized in section 2. Based on that, we build a stack-model that explains their dependencies and interconnections, described in section 3. Section 4 presents the results of an exemplarily application of the model to a knowledge-based system. We conclude the paper with a short summary of the preliminary results and the future prospects in section 5.

2 Tailored Usability Criteria

In this section we shortly introduce the examined usability resources and give reason for their choice. Afterwards, we present the tailored criteria that we reassembled from the existing resources, and we also explain their relevance for the context of knowledge-based systems.

2.1 Resources

The following usability resources have been taken into account: 10 heuristics of Nielsen [11, Ch. 2], 15 heuristics of Muller et al. [10], 11 heuristics of Constantine & Lockwood [4, pp. 45 ff.], 15 heuristics of Kamper [7], 10 heuristics of Sarodnick & Brau [14, pp. 140 f.], 8 principles of Norman [12], 8 principles of Shneiderman [16, pp. 74 f.], 16 principles of Tognazzini [18], 7 principles of DIN ISO 9241-110 [15], and selected principles from Lidwell's "Universal Principles of Design" [9].

Those resources are widely known and applied for usability evaluation, thus constituting today's state-of-the-art. In addition, there exist many more guidelines or styleguides that describe usability criteria in a more detailed manner—for example, Thissen's *Handbook on Screen Design* [17], or Apple's *Human Interface Guidelines* [1]. Such resources, however, are often already much too detailed, which makes them more difficult to tailor adequately for our purpose. This is the reason why we chose the more general heuristics as an initial point for defining our tailored usability criteria.

2.2 The Usability Layers

We now first describe each layer of the stack—i.e. each usability criterion—separately. Thereby we first name the key requirement of each criterion, and then summarize its different aspects in a list. We further give reasons for their relevance in the context of knowledge-based systems. The composition of the stack as a whole, and the interconnections and dependencies of its layers, are further described in section 3.

System Image

→ **Key demand:** *create a tailored and comprehensible system model, according to the system's real-world context*

- apply metaphors, story lines, or visible pictures wherever appropriate
- consider and support users's actual work- and taskflow
- consistency between user expectations and the system
- sequence, structure, and group tasks in a natural/logical way
- break down complex tasks into smaller, more intelligible subtasks
- leave simple tasks also simple within the system

We regard the application of an appropriate system image as highly relevant for knowledge-based systems. Due to the often necessary specific expertise, it is all the more important to facilitate the general usage of the system. Here, a tailored metaphor can considerably ease comprehending the system and its mode of operation. Yet, its tailoring to the real-world context and users is essential as to ensure that it is correctly understood; thereby, also user expectations towards the system should be taken into account. As knowledge-based systems often require complex procedures according to their application domain, simplifying tasks—in breaking them down into more intelligible subtasks—can radically ease the system use and thus enhance its usability. A natural and logical task ordering/grouping according to the real-world context should be provided to further increase the users' understanding of the system; this also includes the demand to leave tasks, that are simple by their nature, also simple within a system.

One notably positive example for a well-implemented system image is the consultation and documentation system for dental findings (used as a case study in Atzmueller et al. [2]). As Figure 2 shows, the model of a tooth diagnosis sheet from the daily dental work context was transferred into the web interface in adopting the numbering and alignment of quadrants (Figure 2, I-IV) and teeth (Figure 2, 11-48). Regarding users from the dental medical domain, such a model eases the system's usage drastically.

Self-Descriptiveness

→ **Key demand:** *ensure transparent actions and meaningful feedback*

- make potential actions and options visible and accessible (transparent)
- always provide feedback within reasonable time
- illustrate which actions caused the current system state, and which actions are suggested next
- messages: clear, unambiguous, and concise; preferably in plain language
- in the case of errors, clearly communicate the problem and suggest steps for recovery
- also use visual means for communication/illustration

Regarding the use case of knowledge-based systems as consultation systems, immediate feedback on the current system status seems to be especially valuable; communicating also minor, intermediate results (solutions), that occur during the consultation process, allows the user to benefit at any time from using the system—even in the case of system errors or quitting the system ahead of time. In the consultation context it is also helpful to illustrate how the current state (the currently derived solution) was achieved, and what actions (questions to answer) are suggested next. Furthermore, potential actions and options should be visible and accessible within the system.

An example, where this aspect has been implemented well, is the d3web.Dialog2 system (Krawczyk [8], shown in Figure 3). This consultation system, based on a question and answer metaphor, makes use of the AJAX-technology to provide immediate feedback in displaying also intermediate results (Figure 3, a) as soon as the user provides input—also, already answered questions are greyed out instantly (Figure 3, b), thus illustrating, which answers have led to the currently derived solution(s); furthermore, the next suggested questions are highlighted through a yellow background (Figure 3, c), and potentially additional follow-up questions are presented at once when necessary.

With respect to the data acquisition use case, making all required actions available should be—as for other kinds of software, too—a matter of course. Additionally highlighting the next suggested actions or input offers a high potential to ease the system usage and thus enhance overall usability. Another key aspect in our context is the appropriate communication prior to or in the case of actually occurring errors; its relevance for knowledge-based systems is discussed in the corresponding section *Coping with Errors* afterwards. Finally, we regard visual means an important requirement to enhance the self-descriptiveness of a knowledge-based system—visualizations are able to present even complex correlations (e.g. the structure of a knowledge base, or navigation support in a consultation system) in a condensed and illustrative way, which in turn improves the users' understanding of the system, and thereby contributes to an enhanced usability.

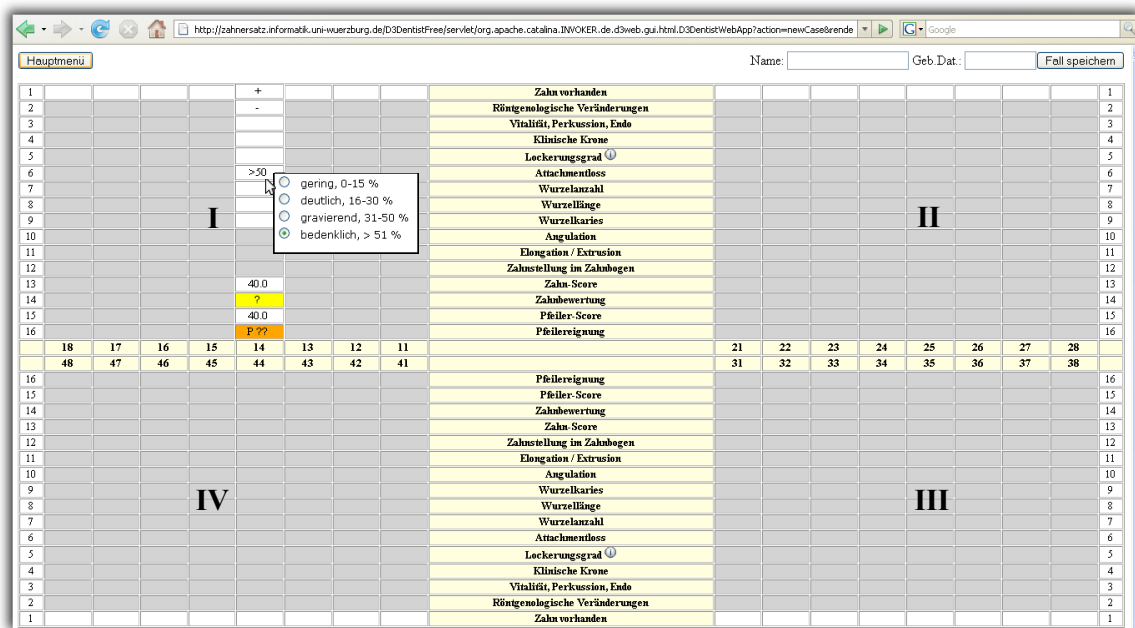


Figure 2: Positive Example for a Tailored System Metaphor, from a Dental Documentation/Consultation System

User Orientation

→ **Key demand:** *implement reasonably constrained user control in a system that is tailored for its potential users*

- provide control through undo & redo, pause & resume
- let users sequence tasks according to their needs
- avoid surprising/awkward system behavior
- deactivate or hide dangerous/critical actions
- constrain critical/complex input
- account for target users' skills, demands, and habits
- speak the users' language

Especially with respect to the often complex tasks occurring in our context, we regard the implementation of constraints, to avoid potential system errors or defective input, as extremely valuable. Critical or dangerous options/actions should be deactivated or completely hidden in the interface. Also pausing and resuming options, as well as the possibility to carry out tasks in any given order, enhance the usability of a knowledge-based system.

Again, the dental documentation and consultation system provides for a good example of reasonably constrained user control. Figure 2 shows that input fields for a tooth are not editable until certain input is provided; this slight constraint assures reasonable input based on previous actions—yet, the user controls in what order to provide the input, once the fields are editable; moreover its the user who generally decides about the order or number of teeth to edit, thus having a great deal of control of this system. The d3web.Dialog2 system (Figure 3), in contrast, provides user guidance through the dialog to a much greater extent—in suggesting the appropriate order of questions, and also in providing follow-up questions straightly dependent on previous input. Yet, in the end the user remains in control also of this dialog, as it is also possible to leave the suggested path and chose a different order of answering the questions.

Apart from a reasonably constrained user control, we also regard tailoring a system to its potential users as highly relevant. Characteristically, knowledge-based systems are

applied in quite specific contexts—thus, for example, medical professional terminology is first-choice for medical staff, whereas this would not be appropriate for biologists. Also, differing user groups often possess different skills and preferred taskflows (e.g. physicians vs. nurses). Tailoring a knowledge-based system to its potential users thus can strongly contribute to its understandability, and consequently to its usability and user efficiency.

Interface Design

→ **Key demand:** *design an aesthetical, yet minimalistic interface*

- hide actions and options not straightly needed from the interface (minimalistic interface)
- clear conveyance and presentation of information, highlighting of important information
- no multi-page displays
- design- and behavior-based consistency
- consistency with standards/conventions from application context
- consider general design guidelines (colors etc.)
- aesthetical, pleasing, "modern" design
- provide distinct "views", where appropriate

Again due to the often complex application domains and procedures related with knowledge-based systems, both the minimalistic and consistent design of an interface are of high priority. Thereby, the former induces simplicity, as important information and required functionalities can be found more easily; the latter, consistency, avoids confusion on the side of the user, which allows for concentration on the task at hand and thus also eases the system usage.

Furthermore, software interfaces should in general be designed in an aesthetic, pleasing way, concerning, for example, the adherence to general design-based principles regarding the choice of colors, fonts, alignment issues etc.

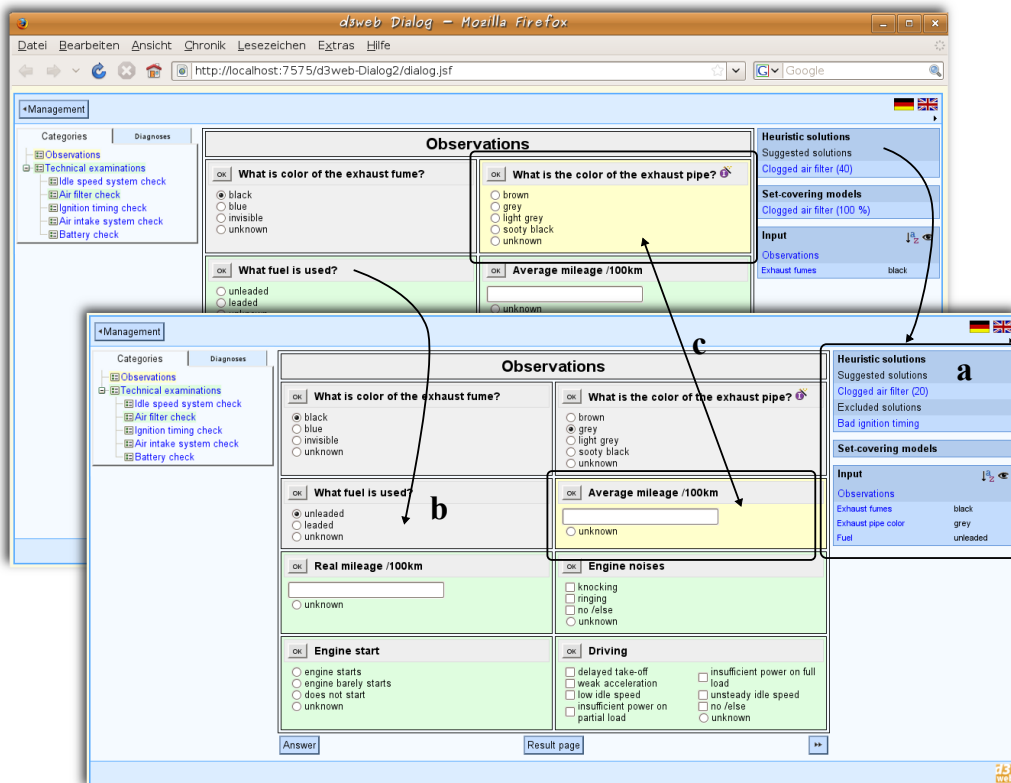


Figure 3: Positive Example for Immediate Feedback, Implemented in a System for Car Fault Diagnosis

This may not directly influence usability, yet it can enhance the user's perception of the system; in turn, a system that is perceived as pleasing is often assumed to be more usable than other systems. In this regard, an interface should as well appear somewhat "up to date"; such interfaces are more likely presumed to be well maintained and tailored for today's hardware. Thus they consolidate the user's confidence towards the system. Concerning the context of knowledge-based systems, we regard the aesthetic aspect also as relevant as it can—not solely but additionally—enhance the user's experience of the system.

Knowledge-based systems often address users from several considerably distinct user-groups—in the case, that those can be clearly specified, we regard the provision of tailored views of the system as highly valuable. Such a view, however, should not only refer to the outer appearance, as e.g. color scheme or alignment. Rather the system as a whole should be affected, thereby considering the specific skills of each user-group. This includes, for example, structuring tasks differently or providing for other kinds of functionalities (e.g. wizard-guided vs. self-guided interfaces).

Coping with Errors

→ **Key demand:** *anticipate and prevent errors whenever possible, in the case of errors provide help for recovery*

- apply appropriate error prevention measures
- hide or deactivate potentially critical/dangerous options and actions
- provide undo & redo for leaving critical states
- concise, comprehensible, informative error messages
- suggest steps for recovery
- handle error internally until problem can be resolved

- prevent data loss

Regarding knowledge-based systems, this aspect also is highly relevant. Those systems generally require a lot of data input, independent of whether they are applied for data acquisition (e.g. setting up the knowledge base) or for consulting (e.g. gathering information to derive solutions)—thus a sudden data loss due to system errors is extremely painful for the users. That case should be prevented through adequate and anticipatory communication (warnings, suggestions of alternative actions or options) and constraints (hiding and deactivation of critical actions and options). Additionally, errors should be prevented through undo & redo facilities, default value presentation, auto-completion, or the intelligent suggestion or highlighting of appropriate next actions. Moreover, also consistency tests should be implemented both on a global and a local basis; examples are limiting the presented questions in a consultation system to the reasonable follow-up questions (global), and verifying critical input to the reasonably processable options (local). If specific critical system states are completely unavoidable, at least tailored, comprehensible error messages should be provided; thereby not only the type and context of the error should be explained, but adequate steps for recovery should be suggested. At any rate, errors should be handled internally without resulting in data loss, or an awkward or surprising system behavior, thus maintaining a stable-running system.

Help & Doc

→ **Key demand:** *provide easily accessible and tailored help and documentation*

- easily accessible at any time and system state
- only as large and detailed as necessary
- searchable index for easier information retrieval

- tailor help & doc to actual system purpose, thus refer to and support actual tasks
- where appropriate, provide tutorials, step-by-step instructions, examples, or reference of the system

It is often stated, that help and documentation are not so important as most people never use them anyway—nevertheless we regard that criterion as a relevant, additional means for improving overall usability in the context of knowledge-based systems. Considering the often tedious tasks and procedures—which cannot always be simplified—an accessible and supportive help- and doc system can be of great value, especially for novice users. Thereby an introductory listing and explanation of best practices or standard tasks regarding the system use can help such users get started. Also a searchable, indexed reference system that covers potential actions and options, specific forms of data input, or that explains more complex parts of the interface can add value to the system. We generally agree, that a system should best be usable without any help at all, yet we consider the specific context as often far too complex to actually completely fulfil this demand; thus, a tailored help & doc system can improve the user's system experience—as it ideally enables the user to solve tasks and even problems on his own—and with that the perceived usability.

3 Composing the Usability Stack

In the previous section 2 we described the different components of the *Usability Stack*. Now we clarify its composition and explain the interconnections and dependencies of the layers. Figure 1 depicts the stack of usability criteria. We begin by explaining the four layers on the lefthand side:

- System Image
- Self-Descriptiveness
- User Orientation
- Interface Design

Layer 1—System Image

We chose *System Image* as the basis of the model because it strongly influences all the other criteria that build upon it; also, it is interdependent with the two additional criteria, depicted as the vertical layers, which will be illustrated in more detail in the subsequent sections. Basically, we not only believe that a tailored model or metaphor can greatly enhance a system's usability, but also, when inappropriately chosen (for the given context or target user group), can turn a system nearly worthless. Moreover we consider the real-world context of a system—as taken into account also within the first layer—an essential basis for the following criteria. Thus we regard the development of an adequate system image as the most important, foundational task when striving for a usable knowledge-based system.

Layer 2—Self-Descriptiveness

An appropriate system image then already contributes greatly to *Self-Descriptiveness*; the former enables users to better predict the system's behaviour, which in turn largely meets the request, that a system should guide users by its design and ability to clearly convey necessary information. The chosen system model or metaphor also influences the appropriate affordances and hints that are to be provided—as useful hints for one context are likely to be

worthless for another context; likewise also necessary actions and options vary depending on the application context or the appropriate task structure. So in summary, self-descriptiveness directly depends on the realization of the system image, which is why we regard it as the second most important criterion for knowledge-based systems and thus as the logical second layer of our model.

Layer 3—User Orientation

User Orientation forms the subsequent third layer within our model. Tailoring a system for its intended target users is highly dependent on the real-world context (as considered within the first layer) that already defines the target users and their specific wants and skills. Also developing and implementing reasonably constricted user control depends on the previous layers. On the one hand, the chosen system image (first layer) often dictates which parts of a system should be controlled by the users and to what extent; on the other hand, this criterion also builds on the findings, which actions and options should be visible at what system state (outcome of the second layer), as only actions/options that are actually visible can be discussed to be constrained or fully controllable at all.

Layer 4—Interface Design

As the three previous layers all influence the final form of the interface, the *Interface Design* layer is stacked on top of the others. First, the chosen system image (first layer) often specifies quite detailed what an appropriate interface looks like as a whole; this is also dependent on the grouping and structuring of tasks. Influencing findings from the second layer are the necessary dialogs and messages required, necessary progress indicators, or the potential need for providing visual forms of navigation support, system overviews, or other kinds of information visualizations. Finally, the analysis of target user groups and their specific wants and needs from the third layer can also affect the interface design: in prescribing, which actions and options should be deactivated for constraining reasons, or if distinct user groups require specific views of the system.

Examining Figure 1 again, however, the *Interface Design* layer builds not only on the lefthand side stacked layers, but also on the vertical layer *Coping with Errors*. This is because we regard it important to first have set up a stable system before thinking about how to present that system to the user. In our opinion, it has first—in the context of the criterion *Coping with Errors*—to be decided, which actions to take to assure the best possible error handling. Based on decisions, as where to display which kind of confirmation or warning/error messages, where to present default values, or where to even hide parts of the interface, an appropriate interface design with tailored dialogs can be created.

In addition to the four layers on the lefthand side, the stack also consists of two criteria that do not built on just exactly one of the other criteria but are interrelated with several ones:

- Coping with Errors
- Help and Doc

In the overview in Figure 1, those two criteria are depicted as vertical layers, interconnected with several other layers of the stack. In the following, they are described in more detail.

Layer 5—Coping with Errors

To begin with *Coping with Errors*, we already pointed out above why this layer constitutes an additional foundation for the fourth stacked layer, interface design. Thus it remains to explain in what ways *Coping with Errors* is dependent or interconnected with the three remaining layers on the left. To begin with the first layer, it is coupled with the fifth criterion as the system model should be designed in a way that it is easily comprehensible by the users and thus reduces potential errors; this also includes the way, how tasks should be broken down (if necessary) and structured. Next, there exists a connection to the second stacked layer, as an important aspect of usable systems is to clearly communicate occurring problems and suggest solutions in the case of errors. Also the demand for suggesting/highlighting the next appropriate actions relates to *Coping with Errors*, as the appropriate implementation also can contribute to error prevention. The third stacked layer finally relates to *Coping with Errors* with respect to the decisions which actions to constrain and to what extent, to prevent severe system errors.

Layer 6—Help & Doc

Help and Documentation finally is a criterion, that in one way or another is related to all of the previously described criteria; thus it is depicted as a vertical layer connected with all other layers on the left. To begin with the system model, yet also the user interface, those should be introduced and explained in some form of documentation; thereby the system's general mode of operation (defined by the system image, first layer) as well as how to conduct specific tasks by pointing out which parts of the interface make available the necessary actions and options (defined by the final interface design, fourth layer) should be explained. Also, easily accessible help—say, in form of a context sensitive help system, or automatically suggesting appropriate actions or problem solutions—can enhance the self-descriptiveness of a system. With regards to the third stacked layer, user orientation, as well as to the criterion coping with errors, a help system documenting existing constraints within the system is valuable in case a user starts wondering why certain actions or options are not always available. Also, help & documentation ideally enable users to solve tasks or even errors on their own, thereby providing them a great sense of mastery of the system.

The final layer, *Usable Knowledge-Based Systems*, building on all the other ones, finally depicts the belief that usable knowledge-based systems can be the outcome of designing—or evaluating and adapting—systems according the *Usability Stack*.

4 Applying the Usability Stack to the Semantic Wiki KnowWE

This section presents the results of exemplarily applying the stack-model to one of the knowledge-based systems recently developed by the Department of Artificial Intelligence and Applied Informatics, University of Würzburg. For this purpose we chose KnowWE, a wiki-based system for collaborative knowledge engineering and knowledge-based consultation (Baumeister et al. [3]). Figure 5 shows the car diagnosis knowledge-base—supporting users in detecting car-related malfunctions—as represented in KnowWE.

4.1 System Image

The system image, provided by KnowWE, is the standard wiki model. Wiki systems are widely spread today and thus their general mode of operation should be known to most people using a computer regularly. This eases getting started with KnowWE at least concerning the standard tasks for potential users of the system, as such tasks—for example, creating a wiki-page—basically conform to the standard wiki way. The use of the built-in consultation features as, for example, the embedded dialog or the in-place-answer lists, are quite intuitive: in the case of the dialog, the user selects the appropriate answers by just clicking on them (Figure 6, a), and in the case of the question sheet, a click on the question category opens a small pop-up containing the answer possibilities (Figure 6, b)—which again can be selected by a click. Once a user starts answering the questions, solutions are derived after each input and displayed in the solutions panel (Figure 5, b). Yet to take advantage of KnowWE's data acquisition features, the particular knowledge markup has to be learned additionally. As KnowWE aims at supporting different forms of knowledge, the markup language has already grown quite complex, and thus probably poses some problems for novice users, or such users that cannot use the system regularly. Some experiments with KnowWE demonstrated that the knowledge-markup, although it is quite easily learnable, still leaves room for potential mistakes.

4.2 Self-Descriptiveness

The second criterion, self-descriptiveness, could so far achieved in parts. First, mostly appropriate feedback concerning the system status is communicated. When using KnowWE for knowledge acquisition—entering and modifying knowledge in a wiki article (Figure 5, a)—the results are immediately visible either when the article is saved or via a preview-feature (Figure 5, d). Concerning the use of KnowWE as consultation system, the recently derived solutions (Figure 5, b) are displayed and updated immediately, according to the latest user input; thus, also intermediate results are accessible. Yet, self-descriptiveness of the system as a whole has to be further improved—basically, the offered feedback is adequate for users that know how to achieve certain goals. Regarding novice users, though, more feedback to get them started—for example, suggesting first actions, or better highlighting the general tasks that can be accomplished—should be provided. Concerning potential errors, in some critical cases warnings are provided—one example is, that a user has to confirm a warning before saving an article, if that page is at the same time being edited by another user. Additional future improvements of self-descriptiveness could be achieved by visualizing the navigation or an overview of the system and its linkages to further enhance the understandability of the system as a whole and thus its usability.

4.3 User Orientation

User orientation is a two-sided issue within KnowWE. On the one hand, the chosen model of a wiki provides for a reasonable user control—the user can decide in person whether to use existing knowledge by just browsing through the pages (consultation), or when and to what extent to enter or modify knowledge (knowledge acquisition). Those two main tasks can also be cancelled at any time ("emergency exits"). To improve this criterion even more, pause and resume actions regarding the data acquisition



Figure 4: BIOLOG Wiki (a Collaboration System for Researchers on Biological Diversity)—Based on KnowWE

task should be considered; this would enable a user not only to completely cancel data entry, but also to resume it from exactly the point he left before and thus not having to start from the beginning. Undo and redo options are currently realized through a version control mechanism that enables users to restore a previous version of the entire page. Using undo while editing a page, though, is not yet implemented so the user has to remember his last edits to be able to undo them manually; once having left the editing mode of a page, undoing the last edits is also not yet possible; still such undo features would be preferable to ease retracting minor mistakes.

The second main aspect of user orientation, however—tailoring the system to particular user groups—is not yet implemented in the basic KnowWE system. The system has so far been designed on a more general level with the goal to be applicable in several contexts and for different user groups. With certain effort, however, it is possible to achieve tailoring to some extent, as the example of the BIOLOG-wiki (see Figure 4)—that is built based on KnowWE—shows. Yet, some tailoring options—for example several kinds of consultation dialogs, or different editor styles—should be included in the basic system as well to enhance user orientation already there, and further ease future tailoring to distinct user groups.

4.4 Interface Design

Interface Design is one aspect of KnowWE that definitely has more attention to be paid to in the future. Although a quite consistent color scheme and appropriate fonts have been applied, the overall aesthetical presentation of KnowWE still needs improvement. Moreover, some parts of the interface should be enhanced with regards to a minimalistic design. The upper right part of the wiki is such an example—in the case that the history of already visited wiki sites contains various objects, the popup of the search-input field partly overlaps the site history and thus contributes to a quite cluttered impression of this part of the interface (Figure 5, c). Another example is the presentation of the actions and options while editing a wiki

page; here, the user is offered the whole palette of editing options and additional features all at once and in too many different ways (buttons, checkboxes, tabbed panes). Another issue with the design of KnowWE is consistency. First, sometimes actions and options are not yet displayed in a consistent way. As Figure 6 depicts, the answer popup for question *fitness* contains an answer-unknown option (represented by -?-)—within the dialog, this option is not given, though. Moreover, as KnowWE is a browser-based application, its appearance could slightly differ from one operating system to another, as well within different browsers. This has to be taken special care of in the future, when further refining the interface. Implemented quite well already, however, is the highlighting of important information—recently derived solutions—in KnowWE by displaying them in a distinct labeled box underneath the navigation menu (Figure 5, c).

4.5 Coping with errors

KnowWE in parts already provides for an anticipation of errors in presenting warnings in the case of critical actions. As already mentioned, if a user tries to save a page that is being edited by a second user at the same time, a warning is displayed to avoid the potentially occurring problems (data loss for the other user, or system problems due to saving the same page at the same time). Yet the user still remains in control—that is, if the user wants to, page saving can be continued after confirming the warning. For improving data safety in general, some kind of sophisticated automatic saving mechanism that takes into account the recent edits of both users could be considered. Another aspect implemented regarding error prevention is an auto-completion mechanism when entering knowledge into the wiki. In the case that, regardless of the auto-completion suggestions, some input has been entered inaccurately, an automatic display of suggestions for the correct input format, or at least an informing message, could be additionally valuable. If an error occurs, at least there exists the possibility of restoring the latest, successfully stored version of the page via the provided version control.

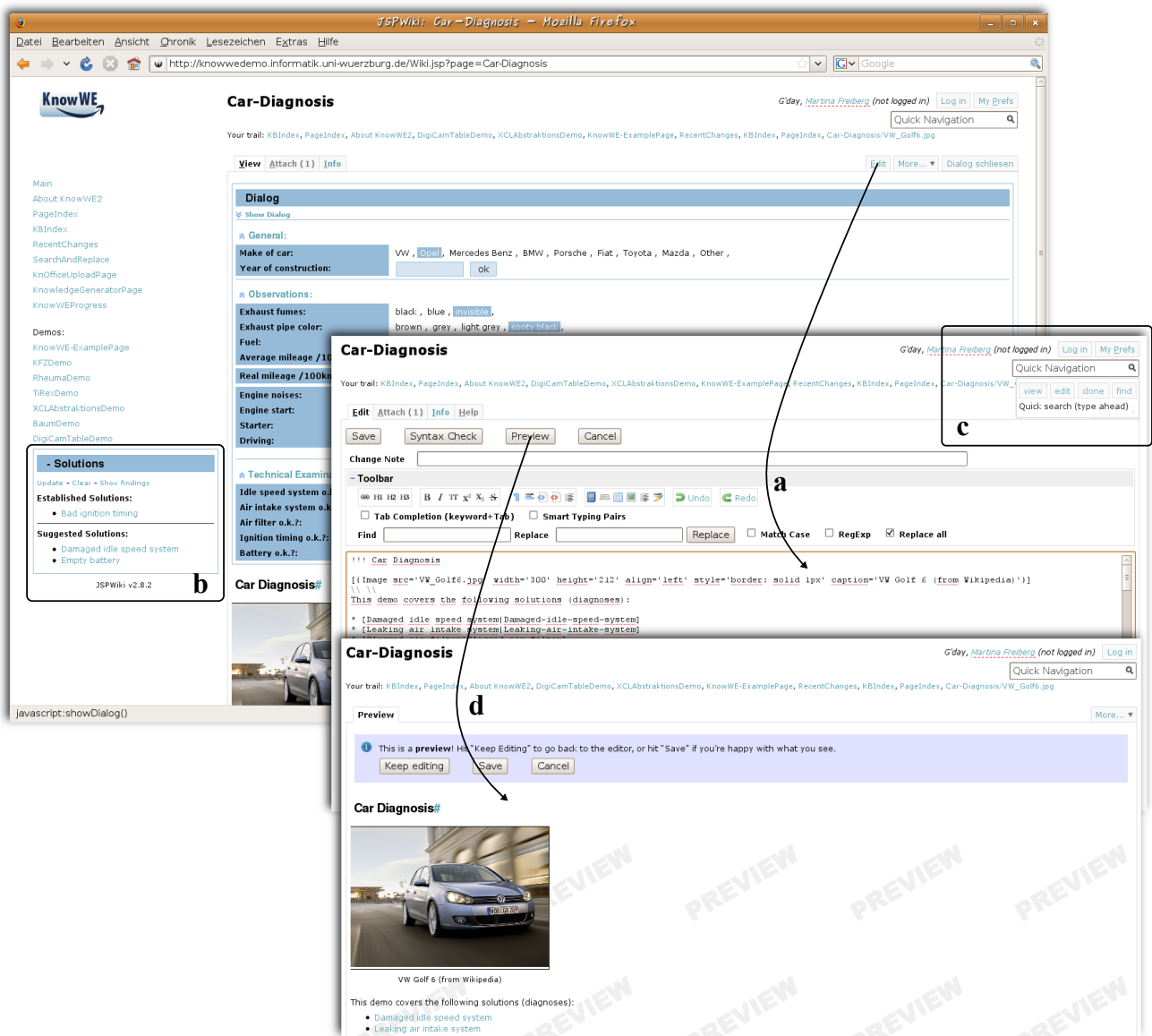


Figure 5: The Knowledge Wiki Environment KnowWE

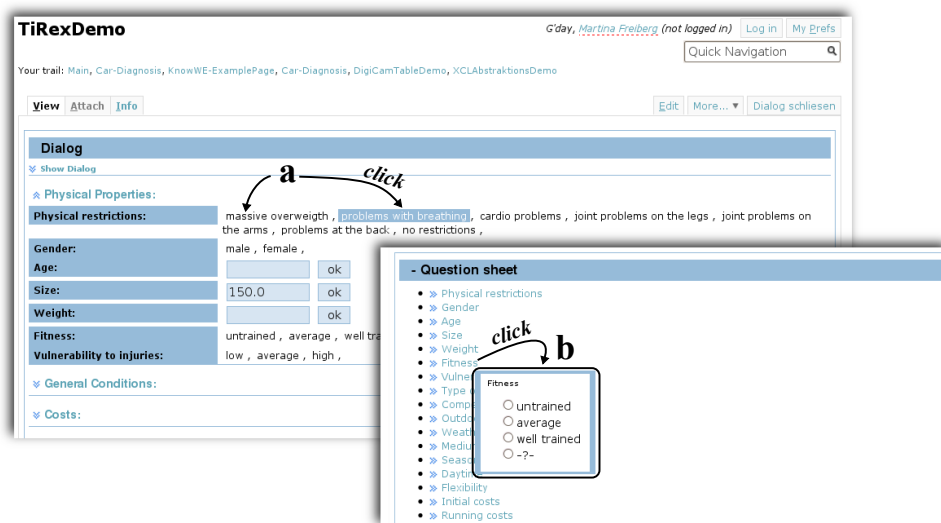


Figure 6: Consultation Features of KnowWE

4.6 Help and Documentation

KnowWE already provides some exemplary documentation; there exists an introductory page that explains how to use the specific kinds of knowledge markup with the help of actual examples. Yet, this existing documentation should be further extended. Potential examples are step-by-step listings of default tasks or of an introductory tutorial, to get the users started initially, or a searchable index of help and documentation topics that also contains links to the actual support sites.

5 Conclusion

In this paper we discussed the need for a more tailored set of usability criteria regarding the design and evaluation of knowledge-based systems. We proposed a novel model—the *Usability Stack* for knowledge-based systems—for summarizing and classifying usability criteria that are relevant in the context of knowledge-based systems.

Some preliminary results concerning the usability evaluation of knowledge-based systems could be gained from analyzing several distinct systems according to a set of usability criteria extracted from known, usability-related literature (described in [6]). This mainly showed the need of creating a new model of tailored usability criteria for knowledge-based systems. Based on those experiences, we reassembled relevant criteria and worked out the model of the *Usability Stack*. Applying the *Usability Stack* to a recently developed knowledge-system, KnowWE, showed the general applicability of the model. Yet it also revealed the need to refine and distinguish aspects of the model with respect to the two main differing use cases of knowledge-based systems, knowledge engineering and consultation. Concerning the particular inspected system, KnowWE, it turned out that some aspects of the usability criteria have been implemented intuitively in the past. Yet, many other aspects that still offer room for improvement could be identified. Thus future work will include the improvement of the KnowWE system by means of the insights gained from this first examination. Thereby, all of the developed criteria will have to be reconsidered; yet special emphasis will be put on self-descriptiveness, user orientation, and interface design, as those have the most deficiencies so far.

Mainly, however, the model will be applied for the evaluation and redesign of other knowledge-based systems; through an ongoing process of applying and refining the model—thereby working out characteristics of the two main use cases knowledge engineering and consultation—we hope to further improve the model, resulting in a detailed catalog of usability criteria and a related process model of their application. Also we hope to gain insights, whether the model should rather be kept generally applicable for different classes of knowledge-based systems with respect to the underlying inference methods, or whether a distinction should be made for those cases.

Finally, we also intent to develop and offer some tool to support and ease the application of the stack-model. Such a tool should provide the elaborated criteria in an easy-to-adapt and easy-to-reuse electronic version—for example, in the form of an electronic checklist. Further it would be interesting to provide such a tool with the ability to preliminarily assess and evaluate a knowledge-based system's level of usability, and provide hints to system developers as to which parts of the system require the most refinements with regards to an overall usability enhancement.

References

- [1] Apple Computer Inc., *Apple Human Interface Guidelines: The Apple Desktop Interface*, Addison-Wesley, 1987.
- [2] M. Atzmueller, J. Baumeister, A. Hemsing, E.-J. Richter, F. Puppe, Subgroup Mining for Interactive Knowledge Refinement, in: *AIME'05: Proceedings of the 10th Conference on Artificial Intelligence in Medicine*, Springer, 2005, pp. 453–462.
- [3] J. Baumeister, J. Reutelschöfer, F. Puppe, KnowWE: Community-Based Knowledge Capture with Knowledge Wikis, in: *K-CAP '07: Proceedings of the 4th international conference on Knowledge capture*, 2007.
- [4] L. L. Constantine, L. A. D. Lockwood, *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*, Addison-Wesley Professional, 1999.
- [5] Frank Puppe and Martin Atzmueller and Georg Buscher and Matthias Huettig and Hardi Lührs and Hans-Peter Buscher, Application and Evaluation of a Medical Knowledge-System in Sonography (SonoConsult), in: *Proc. 18th European Conference on Artificial Intelligence (ECAI 2008)*, accepted, 2008.
- [6] M. Freiberg, Usability assessment of knowledge-based consultation systems, Tech. Rep. 457, Institute of Computer Science, University of Würzburg (2009).
- [7] R. J. Kamper, Extending the Usability of Heuristics for Design and Evaluation: Lead, Follow, and Get Out of the Way, *International Journal of Human-Computer Interaction* 14 (3&4) (2002) 447–462.
- [8] A. Krawczyk, Konfigurierbarer Dialog zur Nutzung von wissensbasierten Systemen, Master's thesis, University of Würzburg, Germany (2007).
- [9] W. Lidwell, K. Holden, J. Butler, *Universal Principles of Design*, Rockport Publishers Inc., 2003.
- [10] M. J. Muller, L. Matheson, C. Page, R. Gallup, Participatory Heuristic Evaluation, *Interactions* 5 (5) (1998) 13–18.
- [11] J. Nielsen, *Heuristic Evaluation*, John Wiley & Sons, Inc, 1994, pp. 25–62.
- [12] D. A. Norman, *The Design of Everyday Things*, The MIT Press, 1988.
- [13] F. Puppe, Knowledge reuse among diagnostic problem solving methods in the shell-kit d3, *International Journal of Human-Computer Studies* 49 (1998) 627–649.
- [14] F. Sarodnick, H. Brau, *Methoden der Usability Evaluation: Wissenschaftliche Grundlagen und praktische Anwendung [Usability Evaluation Techniques—Scientific Fundamentals and Practical Applicability]*, Huber, Bern, 2006.
- [15] W. Schneider, *Ergonomische Gestaltung von Benutzungsschnittstellen: Kommentar zur Grundsatznorm DIN EN ISO 9241-110*, Beuth Verlag, 2008.
- [16] B. Shneiderman, C. Plaisant, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Addison Wesley, 2004.
- [17] F. Thissen, *Screen-Design Handbuch [Handbook on Screen Design]*, Springer-Verlag, 2001.
- [18] B. Tognazzini, *The First Principles of Interaction Design*, retrieved June 25, 2008 from <http://www.asktog.com/basics/firstPrinciples.html>.