User-Centered Consultation by a Society of Agents

Georg Buscher **Joachim Baumeister** **Frank Puppe** **Dietmar Seipel**
Institute of Computer Science
University of Würzburg, Germany
email {buscher, baumeister, puppe, seipel}@informatik.uni-wuerzburg.de

ABSTRACT
User-centered consultation systems can be viewed as systems with trust in the competence of the user, whereas conventional knowledge systems are often based on some mistrust. By offering a variety of formal and informal diagnostic services mediated by flexible agents, the user solves his or her problem by asking the agents for specific information complementary to his or her knowledge. The development process is flexible enough to build minimalist Wiki-like informal systems very quickly, which can be incrementally refined and formalized in a distributed manner. The approach is demonstrated with two case studies.

Categories and Subject Descriptors
I.2.1 Applications and Expert Systems, I.2.11 Distributed Artificial Intelligence, H.3.3 Information Search and Retrieval, Retrieval Models

General Terms
Design, Management

Keywords
Consultation system, diagnosis, agents, Wiki-systems, knowledge-based systems, distributed knowledge acquisition.

1. INTRODUCTION
User-centered consultation systems (UCCS) emphasize the active role of the user during the problem solving process. Especially in domains like medicine, where physicians are highly competent but short in time, users need customized support for solving their problems. Traditional monolithic consultation systems designed to solve diagnostic or therapeutic problems in a systematic manner may be too inflexible for experienced users. A recent evaluation of a medical documentation and diagnosis system in sonography after several years in routine use in a large clinic revealed that standardized documentation was welcomed but that diagnostic help was largely ignored except from beginners [4]. Experienced physicians seem to be reluctant to engage themselves with unrequested advice but have specific questions in special situations. As a consequence, the consultation need is rather inhomogeneous. An economic way to provide this help is to give the user full control over the dialog strategy as well as the level of detail and formality of the knowledge used for consultation. An inherent draw-back is the system's limited capability to detect problematic decisions, because the user can always use shortcuts or jump to a conclusion. Giving the user full control is difficult to achieve with monolithic systems and much easier by a multi-agent paradigm. Here, each agent provides a particular service to the user, and the sum of all agents may cover the competence of a monolithic knowledge system. In such a system, the user may switch from one agent to another by simply following links that are automatically maintained by the system using a Wiki-like technique [6]. The different agent types are organized according to the user requirements for a diagnostic consultation following the well known hypothesize-and-test paradigm of diagnoses. They deliver the following services to the user:

1. Elaborate symptom classes to suspect diagnoses (hypothesize)
2. Clarify suspected diagnoses by checking their symptoms (test)
3. Determine adequate therapies for the identified diagnoses

Symptom classes (e.g. in the medical domain: chest pain, dyspnoe, ECG or blood count) represent a chief complaint or a diagnostic test. A symptom class comprises a set of attributes (e.g. duration of chest pain, aggravating or relieving factors of chest pain). Each attribute has a range of possible values. A pair (attribute, value) is called a symptom (e.g. aggravating factor of chest pain = physical exercise). While symptoms represent the input to the system, diagnoses and/or therapies are the output. Diagnoses are inferred from the symptoms and rated with a probability or probability class like improbable, unclear, possible, probable. In some domains, therapies are tightened to diagnoses (e.g. in many technical domains, diagnoses are defect components and the therapy consists of replacing them) and do not need to be represented by different agents, while in other domains (e.g. medicine) the concepts are distinct, i.e. there is a many-to-many relation between them.

1.1. Architecture
The architecture of a UCCS is built upon global hierarchies for symptom classes, diagnoses and therapies, respectively, i.e. the terms for these concepts are the backbone of the system. For each term, one or more agents can be defined providing a service according to the above mentioned scenarios. The service to the user is provided either by informal agents (i.e. text or hy-

K-CAP'05, October 2–5, 2005, Banff, Alberta, Canada.

permedia) or by different kinds of formal agents, where the user is asked to enter data and the agent infers conclusions. Symptom class agents provide the service to give the differential list of diagnoses according to entered symptoms. In addition, several formal agents for different symptom classes may combine their individual differential diagnoses lists to an integrated one. Diagnosis clarification agents (or clarification agents in short) are able to check a diagnosis' probability (or probability-class) and provide background knowledge about the diagnosis. Of course, informal clarification agents simply present knowledge necessary to rate them; the interpretation must be completely done by the user. Formal clarification agents require data to be entered, but similar to formal symptom class agents, the level of detail of the input data may be determined by the user. Agents for recommendation of therapies are similar to symptom class agents, except that they use diagnoses as input. The distribution of the knowledge in different agents may require the redundant acquisition of knowledge, because in an ideal system, the knowledge of all symptom class agents together is equivalent to that of all diagnosis clarification agents.

1.2. Development Process

The development process of UCCS differs from that of traditional consultation systems. The multi-agent approach allows for a distributed "mass collaboration" [8]. However, we recommend defining the global hierarchies centrally in order to ensure that the agents fit to each other (e.g., that different symptom class agents refer to the same diagnoses). In its simplest version, the various agents of UCCS are built and used similar to Wiki-systems [6], where different authors contribute articles (i.e. informal agents) that are automatically linked by Wikiwords according to terms of the global hierarchies. In its most sophisticated version, formal agents may interact in an automatic way to mimic a monolithic knowledge system (e.g. if a user chooses to do without the user-centered dialog strategy).

The development process for the formal agents (compare [9]) may vary from a completely centralized approach, where a monolithic knowledge base is uploaded and automatically divided into different agents together with generating their communication model, via a semi-local approach with a central standardized terminology for the symptoms inside the symptom classes but decentralized definition of the inference knowledge (providing the possibility to combine the local knowledge bases in a central knowledge base for checking consistency) to a local approach, where each agent is built independently from each other. In the semi-local approach it might be sufficient to define only either the symptom class agents or the diagnosis clarification agents and the other ones are generated automatically. In the local approach, where symptom class and clarification agents are developed independently, the redundancy may cause inconsistencies, e.g. a symptom class agent suspecting a diagnosis, while the corresponding clarification agent does not even mention the symptom class. However, the standard interpretation is, that the clarification agents are more reliable (e.g., if there are cheap tests for verifying a diagnosis, it might not be necessary to mention all symptoms only weakly associated with that diagnosis).

Another issue is that different agents may have different levels of details and/or different formal representations of their knowledge. On the one hand, different levels of detail and different representations may speed up the development process because high costs of knowledge formalization can be limited to important areas of the application or can be incrementally added. On the other hand, such an approach impairs the ability to test the consistency and quality of the included knowledge.

User-centered consultation systems can therefore best be viewed as systems with trust in the competence of the user, whereas conventional knowledge systems are often based on some mistrust. In other words, UCCS are built to complement the competence of the user instead of doubling it. Depending on the assumed competencies, the development process is flexible enough to build minimalist Wiki-like informal systems very quickly, which can be incrementally refined and formalized in a distributed manner.

1.3. Related Work

Collaboratively developed and used knowledge systems were investigated in the past. For example, Brazier et al. [1] also motivate the construction of knowledge systems that allow for a more flexible approach of problem-solving in which "different alternatives [of a solution] are thought through and compared". They identify three basic types of agents within their user-centered design of knowledge systems: the user, the diagnostic support agent and the clarification agent. For the diagnostic agent and the clarification agent only knowledge based on formal task-models is allowed but the user is strongly integrated into the problem-solving process. Typically, the diagnostic agent identifies a list of possible hypotheses that is presented to the user. Consequently, the user investigates the hypotheses and calls the clarification agent when lacks in clarification are determined. Then, the clarification agent generates appropriate explanations for the hypotheses and presents them to the user. If the user agrees with the provided explanation the user gives the selected hypotheses to the diagnostic agent for further reasoning. When compared to the approach presented in this paper we see that the interaction of the user with the system of Brazier et al. is more limited within the given workflow of problem-solving, whereas in the presented approach the user can interact principally in all phases of the problem-solving process. An example for a system based on a more informal approach is the kinesys approach [10], that relies on the collaborative acquisition of a knowledge system mainly used as a knowledge sharing system. Here, only little formal knowledge is included and inference is limited. Similar to the previous system Fuchs-Kittowski and Köhler [2] apply a Wiki-concept for cooperatively building a knowledge base, but also limit their work to informal knowledge. However, the approach explicitly describes the processes of accessing, validating and restructuring knowledge. In summary, the work presented in this paper tries to integrate formal and informal approaches of problem-solving and to provide a flexible integration of the user in this process.

The rest of the paper is organized as follows: In Section 2, we focus on critical issues of UCCS in comparison to traditional knowledge-based consultation systems. Section 3 describes the

user's and developer's view on UCCS as well as its architecture in some detail and Section 4 provides two case studies emphasizing the user's and the developer's view, respectively. Section 5 concludes the paper and gives an outlook for future work.

2. CRITICAL ISSUES

UCCS differ both from the user's and developer's view from traditional knowledge-based consultation systems (TKCS). While the advantages were outlined in the introduction, we have in this section a closer look at the problems. As a typical example of a TKCS in routine use, we compare it with the above mentioned monolithic knowledge-based documentation and diagnosis system for sonography SonoConsult [4].

2.1. Consultation Style

In UCCS, the user is not required to enter the full problem description like in TKCS. Instead he or she may focus on arbitrary subproblems, e.g. is there sonographic evidence for liver cirrhosis? Such a question can be answered based on a few aggregated or based on many detailed symptoms (compare Fig. 3). Again, the user makes the choice. However, this flexibility of UCCS does not fulfill the prerequisites for services of the TKCS SonoConsult like documentation with automatic report generation and data mining or critic functionality by comparing the user's with the system's solution. These services depend on a rather systematic dialog strategy guided by the case peculiarities and not by the needs of the user. Case-directed data collection might be provided by an additional agent. If the necessary data is available, a report generation agent, a critic agent or a subgroup discovery agent could be added. All these agents might reuse knowledge from UCCS-agents, but this topic is beyond the scope of this paper.

2.2. Interaction between Agents

Currently the UCCS-agents are loosely coupled using the Wikiword technique, i.e. users can manually switch to related agents that apply an equivalent subset of terminological entities. The diagnoses inferred by different formal symptom class agents can be merged in a simple manner described in Section 3.1 in more detail. However, a more elaborated agent approach that would allow an extensive transfer and interaction between the agents for solving the current problem stated by the user is possible.

Since all agents in the system are based on a common top-level terminology, e.g. formal agents for clarification could be called automatically to verify the hypotheses generated by symptom class agents. The clarification agent would then ask his questions to the user. In part, the answers may be known already from a symptom class agent. An automatic data exchange would therefore be necessary. It is easy in the semi-local development approach (see section 1.2), but requires special mapping information between semantically similar but syntactically different symptoms in the local approach. Another open issue is to summarize the ratings of different clarification agents concerning the same diagnosis, if they are based on the same symptoms, but use different knowledge representations or were developed by different authors.

2.3. Maintaining Quality and Consistency

In general, the evaluation of knowledge-based consultation systems has been studied extensively in the past, e.g. [12]. Thus, formal methods are available for testing the quality and consistency of monolithically built systems. In contrast, the evaluation of collaboratively built systems is still an open research issue. Although our setting requires a fixed top-level terminology for all agents, the varying level of detail and formality worsens the formal assessment of the entire knowledge.

For evaluating the quality of a UCCS we have to consider 1) the correctness of single agents, 2) the consistency between different agents concerning (parts of) the same topics, and 3) the relevance and usefulness of the agents w.r.t. the user's requirements.

The first issue, i.e. the correctness, can be implemented for formal agents by empirical testing or analytical methods. For informal text agents no formal evaluation can be provided but user rankings of the particular agents can help to substitute a formal evaluation in some kind. The second issue, i.e. the consistency, also is possible for formal agents building on an equivalent terminology, for textual agents user rankings must be applied, analogously. The third issue, i.e. evaluating the usefulness, fully requires the results collected from user rankings and the counted use of the particular agents.

We see that user rankings are an essential method for evaluating and maintaining a UCCS. As a consequence, the architecture provides a simple to use rating facility for each agent and a process model for using the feedback to improve the services of the agent. But even agents with a very high rating may not form a good team. Our solution is to indicate team properties and target groups by metadata of agents (in medicine, e.g. the level of detail; the underlying assumption like school medicine, homeopathy or naturopathy; the target group like laymen or physicians; etc.) and only agents having the same team properties and target groups should be combined by the user. Therefore, the properties and target groups must be transparent to the user.

2.4. Organizing the Community of Developers

The TKCS SonoConsult was developed mainly by one expert. UCCS try to avoid this knowledge acquisition bottleneck by distributed development of the agents. But even if different developers adhere to the same target groups, the agents might be inconsistent and of very different quality. While in Wikisystems like wikepedia [http://wikipedia.org] everyone may change an article and it is hoped, that in the end, a widely accepted position wins recognition, we are skeptical, that this transfers to diagnostic agents, because they interact much more closely than lexicon articles. An alternative is some central control, e.g. by a "knowledge champion" [11], who supervises the distributed development, or by a defined community of developers, which know each other well enough to trust in the development capabilities or by a peer referee system. However, at the moment we have little experience with the relative strengths of those organizational alternatives of distributed development. In any case, the agents have a visible author, who invests his or her "reputation".

3. ASPECTS OF UCCS

In this section, we describe a UCCS with its different functionalities in detail. We distinguish the different views of the user, who intends to consult the system, and the developer, who creates or modifies agents. After discussing these views we describe some interesting aspects of the presented architecture.

3.1. User's View

The consultation system is accessed by a standard web browser. As shown in Figure 1 the user interface mainly consists of two windows.

The first window is the *main frame* of the application. It shows the global hierarchies of the system, which are visualizations of the top-level terminologies, and contains links and buttons directing to agents and knowledge, respectively.

The second one, i.e., the *agent window*, shows the user-selected agents. Depending on the agents' degree of formality, they can be visualized in different manners, some of which are shown in Figures 1-3. Furthermore, the window shows a panel that offers the possibility to rate the quality of the currently running agent, it displays a history about the previously viewed pages and it offers a link to an explanation table, which will be discussed later in this section.

To provide enough space for the contents of the two windows, they are partly covering each other as hinted in Figure 1. However, it is possible to quickly switch between them by a single mouse click.

It is possible that several agents are defined for the same hierarchy element, which means that they are dealing with the same topic and are partly redundant. Then, the user has several alternatives to select the most appropriate agent for a consultation: On the one hand, the user can choose the type of agent he or she likes to view. For example, textual agents may exist providing informative texts. Alternatively, more formal knowledge may be available like agents based on heuristic rules or decision trees. On the other hand, as already mentioned in Section 2.3, the user may select an agent according to its target group. The third alternative is to select agents according to their user ratings. For example, the user may prefer agents with high scores or those that were developed by well-known domain specialists.

As denoted in Section 2.2, the system applies the Wiki-word linking technique, which provides several navigation options automatically. In more detail, the Wiki-word linking technique is based on textual analysis. As a prerequisite, a set P of phrases has to be defined, which have special meanings. On this background, a given text is scanned word by word to detect all of these distinguished phrases. (In the context of Wikis, these phrases are sometimes called Wiki-words.) Finally, the detected phrases will be transformed into links pointing to predefined, phrase-dependent locations. In our case, exactly the elements of all global hierarchies are contained in the set P of distinguished phrases. If such an element e is detected in, e.g., an article of a textual agent, links will be generated pointing to the assigned agents of the element e. Thus, the user is able to switch directly between different agents without having to localize them in the global hierarchies.

The agent rating interface is situated on top of every agent so that the currently displayed agent can easily be rated by the user at any time. Agents are rated by either writing a textual comment or by attaching a numerical mark. Both ratings are assigned to a particular agent's version. Thus, the users may constructively criticize an agent whereupon its author may develop an update.

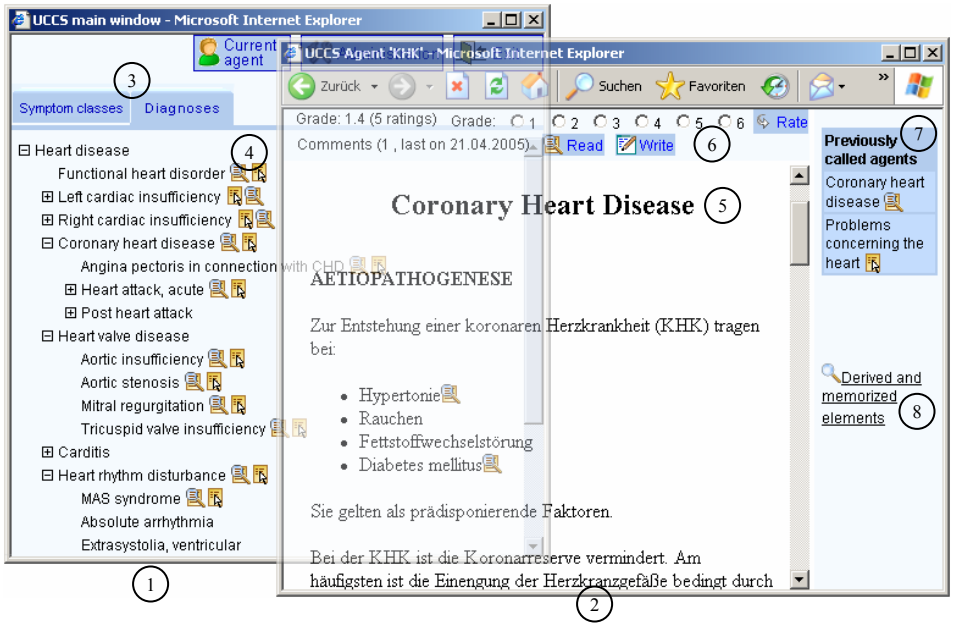

Figure 1: The two main windows of a UCCS, which are partly overlaying each other. The left window (1) shows the global hierarchies (3, currently, the diagnoses-hierarchy is activated). The icons behind a hierarchy element (4) are links to informal (magnifier-icon) or formal (cursor-icon) agents. The window on the right (2) shows the current agent (5), an interface for rating (6), a history of previously called agents (7) and a link to the explanation table (8).

The system provides an explanation table, in which diagnoses that are inferred by different formal symptom class agents are merged. If a formal agent infers a diagnosis, then it will set a score on the diagnosis' account. In general, the scores from different agents will simply be accumulated. However, the results of formal agents that are associated with the same hierarchy element, i.e. symptom class, must not be considered both, because they presumably are founded on the same information. Therefore, the system distinguishes between aggregatable and redundant agents. Redundant are those that belong to the same symptom class; all others are considered aggregatable. In the case of several redun-

dant agents, the system provides different selection strategies for their results: take the result of the agent with the highest reputation (user's rating), consider the agent belonging to a particular target group, or else take the maximum score of all agents. Ideally, only those agents should contribute scores to a diagnosis' account, which belong to the same target group, because in this case it is more likely that they are properly working together. The merging process of different agents' results in the explanation table is only functioning correctly, if the numerical scores that agents set to diagnoses' accounts, are standardized. Therefore, as mentioned in the introduction, we use a scale, which maps scores to categories describing their semantics, i.e. "improbable", "unclear", "possible" and "probable".

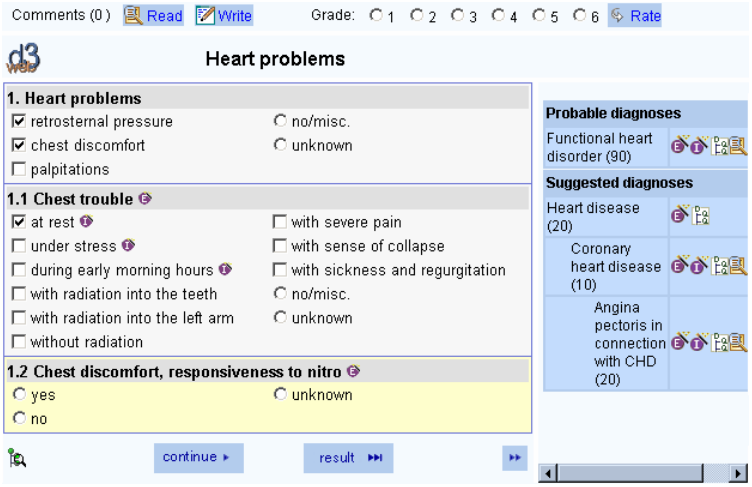

Figure 2: A formal agent suggesting diagnoses according to the indicating sign's specification. The inferred diagnoses are presented on the right. The user has the possibility to call agents associated with the inferred diagnoses by simply clicking on the magnifier-icons behind them.

Process Model for Consulting a UCCS

Before describing typical procedures for using a UCCS, we assume that the user is aware of his or her symptoms and that he or she wants to find a proper explanation, i.e. diagnosis for the given symptoms.

After starting the system's web interface, the hierarchy with symptom classes will be presented on the screen. Then, the user will possibly proceed in one of the following ways to find a feasible diagnosis:

If the user cannot classify the symptoms precisely or only wants to obtain general information about some symptom classes and related diagnoses and therapies, an appropriate way is to select a symptom class with high relevance out of the global hierarchy and to call one of the attached informal agents. Next, the user will be provided with informative texts about the symptoms' characteristics, which are containing Wiki-links pointing to possible diagnoses. After selecting a diagnosis, which seems to be sufficient, the user will take the appropriate linkage to another textual agent providing detailed information about the diagnosis. Until here, this scenario of consultation can be identically implemented by Wiki-systems.

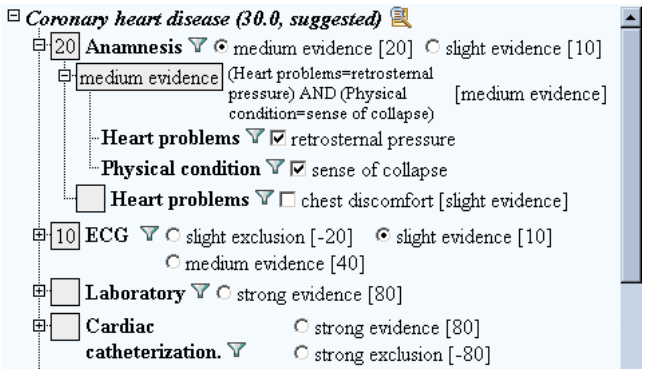

Figure 3: A formal agent to clarify the diagnosis "coronary heart disease". All symptoms that are indices for or against that diagnosis are presented in a tree-like structure and are grouped by examinations that are helpful to clarify the diagnosis. On the one hand, the examinations can be valued directly e.g. by expert users (see ECG). On the other hand, e.g. in case of unsure users, it is possible to put in concrete results for an examination, whose value will then be determined by the system (see Anamnesis). The scores from different examinations are accumulated and set to the diagnosis' account.

A possibly more sophisticated way to find an explaining diagnosis for the given problem includes the possibility to select multiple symptom classes. This comes into consideration, if the user is aware of his or her symptoms in more detail and is able to classify them. For each relevant symptom class the user selects an attached formal symptom class agent that is able to automatically suggest diagnoses (e.g. an agent based on heuristic rules as shown in Figure 2). The agents generate their solutions (i.e. diagnoses with supporting value) independently from each other. The resulting diagnoses of the different symptom class agents are merged in the previously described way and presented to the user. Now, the user selects a diagnosis with a high supporting value and opens the attached informal agent, which provides textual information for a more detailed differentiation. This is done in the same way as it was described in the end of the previous paragraph.

An example for a formal symptom class agent concerning the symptom class "heart problems" is presented in Figure 2. It is a guided dialog, which means that questionnaires are successively presented to the user by taking into account his or her previously provided data. However, it has to be emphasized that it is not necessary to answer all questions. On the contrary, the user can abort the dialog at every time and view the suggested diagnoses that the agent derived so far.

The third typical approach for using a UCCS utilizes formal diagnosis clarification agents. This may be an extension of the second alternative or a completely different approach. Here, the user calls a clarification agent for a diagnosis. The clarification agent, e.g. presented as a clarification tree, will show all relevant information to rate the diagnosis. The agent will also notice these indicating signs, which the user did not consider in his or

her initial symptom class selection, but which are yet relevant to decide the diagnosis.

An example for a clarification tree is presented in Figure 3. The advantage of such a presentation is that the user can input data at different levels of abstraction, so that he or she is able to choose the appropriate level of detail, according to his or her degree of expertise. The root of the tree is the diagnosis the user wants to clarify. Especially in the medical domain it is reasonable to arrange abstract nodes for relevant examinations and tests on the next level. These abstract nodes may themselves depend on more concrete symptoms. Therefore, expert users have the possibility to value the results of examinations on the whole without going into detail, while less experienced users have the option to put in concrete results concerning an examination and let the system infer its degree of evidence. According to the degree of evidence, the examination nodes are scoring on the diagnosis' account positively or negatively. The scores from different examination nodes are accumulated.

3.2. Developer's View

Developers have two possibilities to modify a system: First, they can extend the global hierarchies by adding new entries, i.e. symptom classes, diagnoses or therapies, and second, they can add new agents or change existing ones. For both possibilities, user interfaces are provided, which can be accessed through a standard web browser.

According to a "guided" strategy to build up a new UCCS, the global hierarchies have to be mostly predefined by the initiator. As we expect typical applications for our type of consultation system in the medical domain, where standardized ontology-like structures (e.g. ICD10 standard for medical diagnoses) are already available and generally accepted, the predefinition of hierarchies is a pragmatic way to provide a substantial fundament for a new system. Especially in medical systems, we do not expect the necessity of elementary modifications of the hierarchy structures after the initial setup. Therefore, only the addition of new elements is allowed, in contrast to other types of modification. However, for other domains, this assumption may not hold. In this case, there will be a higher initial effort to create a generally accepted structure of the global hierarchies.

The import process of a new agent into a UCCS firstly includes a description of the new agent's metadata and secondly comprises a specification of its type-dependent knowledge. The former is identical for every type of agent while the latter differs.

The metadata of every agent consists of the following entries:

- Identification (ID) of the agent, including a unique name, the current version, and a comment concerning the current version.
- Data related to the agent's author, e.g., the developer's name.
- A description of the agent's competencies. First, this includes references to elements of the global hierarchies, i.e. symptom classes, diagnoses or therapies that the agent is assigned to. Secondly, memberships for target groups that were motivated in Section 2.3 may optionally be specified here.
- In case of formal agents: A sub-terminology, that specifies import and export data that the agent is able to share with other agents. In contrast to the top-level terminologies, such sub-terminologies especially provide a detailed description of symptoms with discrete values that are belonging to a symptom class.
- Data specifying the agent's type of knowledge (textual, heuristic rules, conditional probabilities, …). The actual knowledge is not included here.

The agent's knowledge has to be specified additionally according to the agent's type. Formal agents have to be specified in a format, which can be interpreted by one of the system's reasoning engines (see also Section 3.3). For instance, to create a textual agent, pages have to be provided in HTML-format; to create a formal symptom class or diagnosis clarification agent, a d3web knowledge base [7] has to be developed. In all cases, the provided knowledge can be imported online into the consultation system using a web-based interface.

Thus, the presented architecture provides the possibility for distributed development: Any developer can create an agent of any possible type and import it into the system. The degree of cooperation between a newly imported agent and the already existing agents can be defined by sub-terminologies specifying import and export data, which is shared among them.

3.3. System Architecture

The system architecture consists of five different main parts as depicted in Figure 4: the UCCS interface, GUI components, reasoning engines, a Wiki-word module and a repository containing agents and terminologies. The first four parts are implemented using the programming language Java, while the terminologies are written in XML.

The UCCS interface is a module, which directly communicates with the user's and developer's side by taking their HTTP requests and sending HTML responses. Its main responsibility lies in generating most of the user interface, in providing agent-import interfaces for developers and in managing agents and terminologies. For example, it views the global hierarchies, it

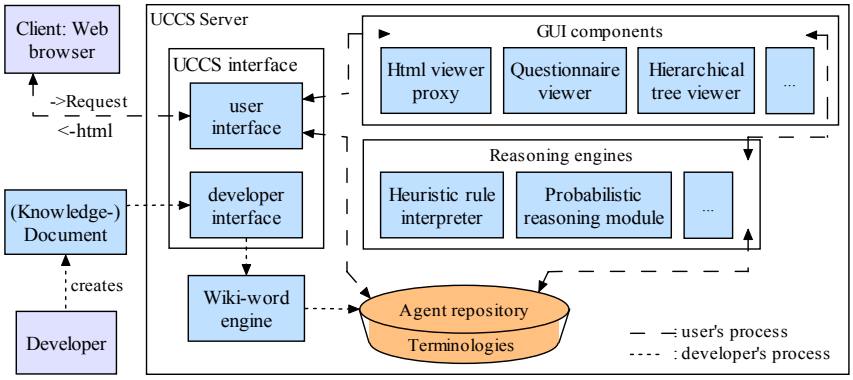

Figure 4: Architecture of a UCCS.

updates the agent history while the user is performing a consultation and it is responsible for managing agent-ratings and the explanation table.

There are different kinds of GUI components, which are used to view the different types of formal agents in an adequate form. Figure 2 shows an agent based on heuristic rules presented by the questionnaire viewer component. The hierarchical tree viewer component is applied in Figure 3 to view a clarification agent also based on heuristic rules.

Each of the reasoning engines is responsible for interpreting a different kind of formal knowledge. All of the reasoning engines are independent from each other and determine the kinds of knowledge of formal agents, which can be accepted and imported into the system. For example, there is a heuristic rule interpreter based on the expert system shell d3web [7], which is used for formal symptom class and diagnosis clarification agents. To include further types of reasoning engines, they have to be integrated directly by utilizing the Java programming language and by providing predefined interfaces for input (user data) and output (solutions). However, the integration of a new reasoning engine has to be done by a system designer rather than by a developer (by means of Section 3.2).

As already mentioned, the system utilizes centralized top-level terminologies defining all hierarchy elements, to which agents may be associated. Every global hierarchy (symptom classes, diagnoses, …) is basing on a different terminology. As sketched in Section 3.2, sub-terminologies can be added to hierarchy elements and extended arbitrarily by the developers. Internally, they are stored separately and do not modify the top-level terminologies, which therefore remain unchanged. In addition to the terminologies' purpose to provide determined hierarchal structures, they also are the central basis for data interchange between different formal agents. The intention is that the user will never be asked twice for the same pieces of information by different agents. If a formal agent requires some information from the user, it is demanded to check a terminology-associated blackboard first.

A centralized agent repository contains all agents that are available in a system. The agents are stored as jar-compressed files containing all necessary information, i.e. metadata and type-dependent knowledge. Therefore, simply the presence or absence of a jar-compressed file determines, whether the appropriate agent is available in the system or not.

4. CASE STUDIES

We provide two case studies: one to describe the development process of UCCS and another one to show the usability of the system from the user's side. The obtained results were very encouraging.

4.1. Case Study for Development Process

To examine the development process, four different distributed knowledge bases were developed in a student project in winter term 2004/2005. The students had the task to build up new UCCS a) for movie selection, b) pub selection in Würzburg, Germany, c) rock classification and d) medical diagnosis for layman. The first, second and fourth distributed knowledge bases were developed by about ten students each; the third one by only two students. Basically, the same process model was applied in each case.

The first step was to define all required global hierarchies which implied a decision about the intended scope of the system. The students listed the movies, pubs, rocks and diseases respectively in diagnosis hierarchies and decided about the symptom classes, e.g. for pub selection: prices, selection of beverages, style and sound intensity of music, atmosphere, etc. This step was rather straightforward, although some extensions were made later.

In the next step, the diagnoses and symptom classes were divided among the students and each student had to write textual articles including pictures and links for their agents. For symptom class agents this meant to refine the symptom classes to concrete symptoms indicating diagnoses, and for diagnosis clarification agents, background information and its main symptoms were stated. Since the terminology of the global hierarchies was fixed, there was some coherence between the articles, and the Wiki-word engine linked the articles to each other. While this worked fine for the diagnoses of all domains and for the symptom classes in the medical diagnosis and rock classification domain, the movie and pub selection groups preferred a more formalized approach using a kind of matrix, describing which symptoms indicate which movie or pub. They needed formal agents immediately, while the other two groups used them as supplement to their informal agents.

As previously stated, the architecture of UCCS does not imply a certain representation for formal knowledge. Therefore the matrices could be entered with different editors and interpreted by different reasoning engines e.g. heuristic rule interpreters. In this project, the diagnostic shell d3web [7] was used. Although the original intention behind its architecture was a centralized development process, several plug-ins allow the semi-local approach mentioned in the introduction. The students divided the work to fill out the entries in the large matrix between all symptoms and diagnoses (i.e. movies and pubs). They agreed to stick to a common terminology of the symptoms (remember, that the UCCS architecture requires a shared terminology for the diagnoses and symptom classes, not necessarily for the symptoms), which allowed to integrate the individual contributions in one large d3web knowledge base avoiding consistency and redundancy problems. This knowledge base was distributed to the different UCCS-agents automatically after upload.

Thus the pub and movie group had formalized agents for the symptom classes and both textual and formalized agents for the diagnoses. The rock and medical diagnoses groups had textual agents for the symptom classes and diagnoses and only reluctantly added formalized agents, because they felt not to be competent enough (especially the medical diagnosis group, but the rock classification had also difficulties in knowledge formalization – well known to all participants of the Sisyphus III project [3]).

The intention of the next step was to enhance the quality of contents provided by the students. Therefore, the participants had

the task to rate agents using the UCCS rating functionality by setting grades and by writing constructive comments. Thereupon, they were requested to improve their articles according to the given comments from other participants. Thus, an iterative community-based development was achieved.

4.2. Case Study for Consultation Process

The aim of a further case study is to determine the eligibility of a UCCS from the point of medical student users in a problem-based learning scenario. Due to a new curriculum at German medical schools, case studies are presented to the students rather early in their studies, and if they don't have the necessary background knowledge, they have to investigate it. Our idea is to offer a UCCS for these investigations. Since case studies are often presented as computer based training cases [5], an integration of both system types was an obvious idea.

Since the UCCS for the students needs to be rather comprehensive, reuse of existing knowledge is a necessity. For the textual agents, we have reused and transformed a large HTML-based internet reference site called MediBook [http://www.medicoconsult.de/MT-book/1_inhalt.php], which was enhanced with formal knowledge from an existing knowledge-based system built to generate training cases. The examples in Figures 1-3 are taken form this project. If the students are in the process of solving a training case presented by the computer, then they can ask both very general and very specific questions to the UCCS, e.g. looking for diagnostic hypotheses for particular symptoms presented in the training case or checking, what tests are indicated to validate certain diagnoses or balance reasons pro and contra a hypothesized diagnosis. The UCCS interface allowed to investigate these kinds of questions at different levels of formality and granularity. Experiments showed the usefulness of the UCCS for supporting problem based learning.

5. CONCLUSION AND OUTLOOK

In this paper, we presented the concept of a user-centered consultation system (UCCS) overcoming some limitations of traditional monolithic knowledge systems by assigning a more active role during problem-solving to the user. Thus, a UCCS offers a collection of alternative ways for solving a specific problem by providing different types of agents at varying levels of formality. Based on a multi-agent approach these agents represent autonomous instances for solving a specific problem but can also be used in conjunction for interactively deriving a solution. We described the architecture and the development process of a UCCS and discussed open issues that we are currently working on. Furthermore, we demonstrated the feasibility and usability of the UCCS approach by some case studies that we have completed so far; we reported a medical real-world application in the context of problem-based learning.

In the future we plan a more elaborated approach for the interaction between the included agents. Then, a more detailed shared terminology of the knowledge representation used by the agents will be necessary in order to allow for a formal analysis and integration of the solutions derived by the particular agents. Furthermore, we hope to collect experiences with the ongoing medical application that will expose the usability of the approaches for increasing the motivation for building and maintaining agents, and for rating existing agents.

REFERENCES

[1] Brazier, F., Jonker, C., Treur, J., Wijngaards, N.: On the Use of Shared Task Models in Knowledge Acquisition, Strategic User Interaction and Clarification Agents, International Journal of Human-Computer Studies 52 (1), 77-110, 2000

[2] Fuchs-Kittowski, F., Köhler, A.: Knowledge Creating Communities in the Context of Work Processes, SIGGROUP Bull., 23 (3), ACM Press, 8-13, 2002

[3] Gappa, U. and Puppe, F.: A Study in Knowledge Acquisition - Experiences from the Sisyphus III Experiment for Rock Classification, Proc. of KAW-98: 12th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, 1998 http://ksi.cpsc.ucalgary.ca/KAW/KAW98/gappa/

[4] Huettig, M., Buscher, G., Menzel, M., Scheppach, W., Puppe, F., Buscher, H.-P.: A Diagnostic Expert System for Structured Reports, Quality, Assessment, and Training of Residents in Sonography: Medizinische Klinik 99, 117-122, 2004

[5] Kraemer, D., Reimer, S., Hörnlein, A., Betz, C., Puppe, F., Kneitz, K.: Evaluation of a novel case-based Training Program (d3web.Train) in Hematology, Annals of Haematology, (in press) 2005

[6] Leuf, B. and Cunnigham, W.: The Wiki Way. Quick Collaboration on the Web. Addison Wesley, 2001

[7] Puppe, F.: Knowledge Reuse among Diagnostic Problem Solving Methods in the Shell-Kit D3, in: International Journal of Human-Computer Studies 49, 627-649, 1998

[8] Richardson, M., Domingos, P.: Building Large Knowledge Bases by Mass Collaboration, Proc. K-CAP 2003, 2003

[9] Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W., Wielinga, B.: Knowledge Engineering and Management; The CommonKADS Methology, The MIT Press, 2000

[10] Slodzian, A.: Kinesys, A Participative Approach to the Design of Knowledge Systems, 12th Intl. Conf. EKAW 2000, Springer, LNAI 1937, 435-448, 2000

[11] Smith, R. and Farquhar, A.: The Road Ahead for Knowledge Management: An AI Perspective, AI-Magazine 21, No. 4, 17-40, 2000

[12] Vermesan, A., Coenen, F.: Validation and Verification of Knowledge Based Systems – Theory, Tools and Practice, Kluwer, 1999