Modeling of Diagnostic Guideline Knowledge in Semantic Wikis

Reinhard Hatko, Jochen Reutelshoefer, Joachim Baumeister, and Frank Puppe
Institute of Computer Science, University of Würzburg, Germany
{lastname}@informatik.uni-wuerzburg.de

Abstract

Knowledge acquisition for diagnostic knowledge systems is a complex and tedious task. In particular, the formalization of diagnostic guideline knowledge is challenging for the contributing domain specialists. In this paper, we introduce the formal representation language DiaFlux, that is simple and easy to use on the one hand. On the other hand it allows for the definition of executable clinical protocols, that can solve valuable tasks being executed in the clinical context. Further, we describe a wiki-driven development process using the stepwise formalization and allowing for almost self-acquisition by the domain specialists. The applicability of the approach is demonstrated by a project developing a protocol for sepsis diagnosis and treatment by a collaboration of clinicians.

This paper is currently under submission at the Open Knowledge Models Workshop at EKAW 2010.

1 Introduction

In recent years, knowledge engineering research has been heavily influenced by the emergence of Web 2.0 applications, such as wikis, blogs, and tagging systems. They provide a simplified access and a light-weight approach for knowledge acquisition. Furthermore, those systems usually allow for a distributed and (often) collaborative development process. One of the most popular examples is the wide-spread use of *wikis* as flexible knowledge management tools, both in personal life and business environments. In contrast to standard web applications the content of a wiki page can be created and modified by clicking an (often mandatory) *edit* button located on the web page. Due to the simple markup (less verbose than HTML), users are capable to author the content easily. Wikipedia is certainly the most popular example, where informal *world knowledge* is created and updated by a wiki. Introducing the *semantic interpretation* of wikis, the development of Semantic Wikis [1] allows for a more formalized definition of the knowledge. Today, Semantic Wikis are mainly used for collaborative ontology development, by providing a flexible, web-based interface to build semantic applications.

The main benefit of Semantic Wikis is their possibility to interweave different formalization types of knowledge in the same context. That way, ontological concept definitions are mixed with free text and images within the wiki articles. Such tacit knowledge often serves as documentation of the development process or as pursuing additional information not representable in a more formal manner. In detail, we call the interweaving of implicit and formal elements of knowledge and their interaction in the knowledge engineering process the *knowledge formalization continuum* [2]. The knowledge formalization continuum emphasizes that usable knowledge ranges from very informal representations—such as text and images—to very explicit representations—such as logic formulas or consistency-based models. The metaphor frees the domain specialists and knowledge engineers to commit to a particular knowledge formalization at an early stage of the development project, but offers a versatile understanding of the formalization process. Present Semantic Wiki implementations serve as ontology engineering tools with some support of rules, thus covering a wide range of the knowledge formalization continuum.

In this paper, we introduce the Semantic Wiki KnowWE, that was designed to build decision-support systems, and we propose the graphical language DiaFlux, for modeling of clinical protocols: The contributions of this language are its simple application for developing decision-support systems, since it only provides a limited number of intuitive language elements. Due to its simplicity it is possible to be used by domain specialists and thus ease the application in the knowledge engineering process. Albeit its simplicity, a rich set of diagnostic elements can be integrated into the language, that are required to build sophisticated (medical) knowledge bases. Furthermore, the language allows for the incorporation of less explicit knowledge elements when needed, and thus follows the ideas of the knowledge formalization continuum. To allow for comfortable development of DiaFlux models, we introduce a visual editor integrated into the Semantic Wiki KnowWE.

The rest of the paper is organized as follows: Section 2 briefly introduces the Semantic Wiki KnowWE and discusses the integration of strong problem-solving knowledge into the context of a Semantic Wiki. Also, some knowledge engineering aspects, such as the organization of a distributed knowledge base over a wiki, are discussed. The procedural knowledge representation language DiaFlux is introduced in Section 3. Also the integration of the language into the Semantic Wiki and development process including stepwise formalization is described. Currently, the approach is evaluated by the development of a medical decision-support system. We describe the essentials of this case study in Section 4. The paper is summarized and concluded in Section 5, also giving an outlook for future work.

2 KnowWE in a Nutshell

Wikis became famous as a successful means for knowledge aggregation in an evolutionary 'self-organizing' way by possibly large and open communities without detailed project management about contributions. Semantic Wikis extend the wiki approach by adding formal representations of the wiki content. This extension allows for two different perspectives of the use of Semantic Wikis [3]:

- **Knowledge formalization for wiki:** Here, the informal content of the wiki is in the foreground. The formalization used to help organization, navigation, and presentation of the content. Formal allowing for semantic navigation and search or the generation of new aggregated views on content, e.g., by inline queries.
- **Wiki for knowledge formalization:** In this case, the formalized knowledge base is the goal of the application. The knowledge base is created by the use of the well-known wiki authoring metaphor. One example is the use of a Semantic Wiki as a collaborative ontology engineering tool. The informal content here is used as description and documentation of the formal concepts and relations.

The wiki-based knowledge engineering approach described as well as the system introduced in this paper clearly focus on the latter perspective: The wiki is used as a tool for creating and maintaining formal concepts, formal relations, and informal knowledge containing description as documentation for these. While many Semantic Wikis provide means to create and populate light-weight ontologies, the approach can be generalized to create any kind of formal knowledge bases, e.g., for decision-support systems. The Semantic Wiki KnowWE provides methods to capture and execute problem-solving knowledge. Therefore, a problem-solving layer has been included on top of the ontological layer, defining findings that describe the currently running problem-solving session. KnowWE is designed showing only minimal modifications with respect to look and feel allowing for "backward compatible" use like a normal wiki. Figure 1 shows an article about *Bad Ignition Timing* as part of a car-diagnosis example wiki. It contains textual descriptions of the concept forming a solution of the diagnostic wiki knowledge base. Embedded in this informal content, it contains rules (1) defined by specific rule markup forming the formal knowledge deriving the concept *Clogged Air Filter* as solution of the problem-solving session. Further, KnowWE provides means for testing the knowledge, e.g., by answering popups (2) that are defined in the page content. The status of the derived solutions of the current problem-solving session is shown in the left panel (3). At the top of the formal knowledge block different icons provide additional features like starting a problem-solving session as a guided interview in an external dialog frame or download of the generated knowledge base (4). The problem-solving knowledge (rules in this example) is entered in textual format by specified markup using the standard wiki edit view, which is processed by the wiki engine and translated to an executable knowledge representation. Each time the content is edited, with the page save action the formal knowledge sections are processed by the wiki engine, updating the executable knowledge base accordingly. Different knowledge formalization patterns (e.g., heuristic rules, decision-trees, set-covering knowledge) are supported by the system and can be captured by the use of various markups [4].

Following the *freedom of structuring* principle proposed by wikis, the system does not put up any constraints where formal knowledge should be defined. Objects and formal relations can be used on any wiki page at any location using the markup defined by the current system settings. KnowWE also provides components to test the current version of a knowledge base. For automated testing of the knowledge base behavior, KnowWE also allows for the definition for test-cases, which can be executed after knowledge base modifications. For the creation of variants of a knowledge base the system provides a flexible include mechanism allowing to include arbitrary knowledge elements into a new page forming a new compilation of the overall knowledge corpus (e.g., for fault diagnosis in deviating production series in technical domains).

Wiki-based Knowledge Formalization The major strength of wikis is the general low barrier for contribution due to its simple editing mechanism. However, the definition of a formal knowledge base using textual markups is a complicated task. Autonomous contribution by domain specialists still can be achieved using a stepwise formalization process. Employing the metaphor of the knowledge formalization continuum at first informal knowledge describing the domain knowledge is inserted into the wiki. This can be done by domain specialists not demanding special knowledge engineering experiences. For this purpose also already existing documents can be imported into the wiki as startup knowledge that can be refined manually. After short training sessions discussing and modifying example knowledge bases the knowledge formalization task is started as an incremental process: The domain specialists, not completely familiar with the provided markups and the formalisms, at first formulate the knowledge that is right now only given as informal description, in pseudo-code style inspired by the markup. In cooperative sessions with the knowledge engineers this knowledge is discussed and transformed into correct syntactical shape. This proceeding allows for autonomous contributions, although if not yet completely formalized, by the domain specialists. The web-based collaborative access provided by wikis supports this evolutionary process.

3 DiaFlux - Modeling Clinical Care Processes in Semantic Wikis

This section first describes our application scenario, then a short insight about guideline models in the diagnostics domain is given. Following, we introduce our representation language for clinical protocols, called *DiaFlux* .

3.1 Application Scenario

Clinical guidelines have shown their benefits by providing standardized treatment based on evidence-based medicine. Many textual guidelines are readily available and also shared through the internet, but rely on the proper application by the clinician during the actual care process. While clinical guidelines are mostly textual documents, clinical protocols are an implementation of them, offering a more specific procedure for diagnosis and treatment in a given clinical context [5]. Much effort has been put into the development of formal models for computer-interpretable guidelines (CIGs). Clinical decision-support systems that execute CIGs support the clinician in his decision-making

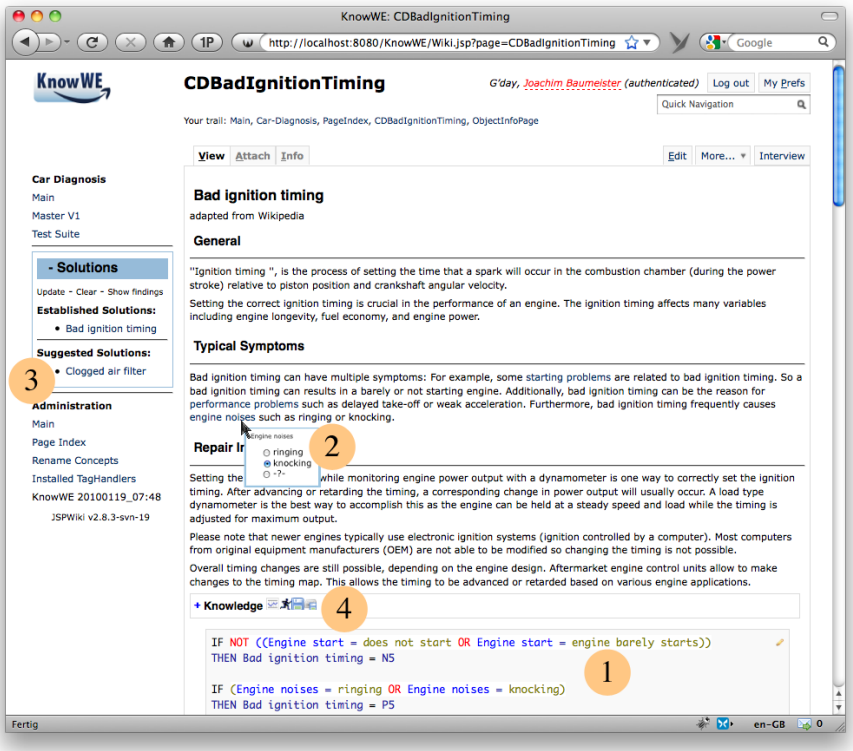

Figure 1: A wiki page of the car diagnosis example wiki containing formal and informal knowledge about the concept *Bad Ignition Timing*.

at the point of care. In the variety of CIG models, each has its own focus, e.g. GLIF [6] focuses on the shareability of guidelines between various institutions, while PRO*forma* [7] focuses on assisting patient care through active decision support [8].

The work presented in this paper is conducted within the project "CliWE - Clinical Wiki Environments"[1]. We investigate languages, tools and methodologies to collaboratively build CIGs by domain specialists themselves. The requirement concerning the language is the development of an explicit and executable representation of diagnostic knowledge for active decision-support systems. Furthermore, we create a development process for simple and effective knowledge acquisition by domain specialists. Finally, the completed knowledge bases will be exported into mixed-initiative systems, that cooperate with the clinical user during the care process.

3.2 Modeling Clinical Processes

For the representation of the tasks, that have to be carried out during the process of care, guideline languages employ different kinds of Task Network Models [9]. They describe decisions, actions and constraints about their ordering in a guideline plan. Often, flowcharts are the underlying formalism to explicitly express control flow, at least at an abstract level. GLIF for example, takes a multi-level approach, on three levels of abstraction, a *conceptual*, a *computable* and an *implementable level*. The conceptual level is modeled using flowcharts for interpretation by humans, but can not be executed by decision-support systems. The formal specification is provided at the next lower - the computable - level. On the implementable level a guideline can be tailored to a specific institution, e.g. by the definition of mappings to patient information systems.

For the specification of a clinical protocol, two kinds of knowledge have to be effectively combined, namely declarative and procedural knowledge. While the declarative part encompasses the facts and their relationships, the procedural one reflects the knowledge about how to perform a task, i.e. deciding which action to take next. In diagnostics the declarative knowledge particularly consists of the terminology, i.e., findings, solutions, and sometimes also treatments and their interrelation. The procedural knowledge for diagnostics in a given domain is responsible for the decision which action to perform next (e.g. in patient care: asking a question or carrying out a test). Each of these actions has a cost (e.g. monetary or associated risk) and a benefit (for establishing or excluding currently considered solutions) associated with it. Therefore, the choice of an appropriate sequence of actions is mandatory for efficient diagnosis and treatment.

We therefore propose a knowledge representation for clinical protocols called *DiaFlux* . It is designed to be a sufficiently expressive knowledge representation for executable clinical protocols, yet intuitive enough for the self-acquisition by domain specialists.

3.3 Modeling with DiaFlux

This paper introduces the formalization of clinical protocols with *DiaFlux* . By combining well-known elements from flowcharts (i.e. nodes and edges) with the tasks that are carried out during diagnostic problem-solving, it allows for a uniform and intuitive acquisition of declarative and procedural knowledge. As known from flowcharts, the nodes represent actions, that have to be executed, and the edges connecting those nodes the order of the execution.

[1]Funded by Drägerwerk AG & Co. KGaA, Lübeck, Germany, 2009-2011.

An edge can be guarded by a condition, that controls the transition to the subsequent node. The nodes represent actions like performing a test or evaluating a diagnosis. The guards are attached to edges and they control the process by checking the state of the declarative knowledge, e.g. the value of a finding or the evaluation of a solution.

Clinical protocols, that are formalized with DiaFlux, are intuitively understandable and more easily maintainable, as the sequence of actions is made explicit. Furthermore, due to the semantics of an underlying application ontology it directly is executable.

Design Goals

When designing a formalism – especially when aimed at being used by non-computer-scientists – a trade-off has to be made between its *expressiveness* and its *usability/understandability*. In our approach we favor usability over expressiveness, offering a minimal set of primitives, though expressive enough for the targeted scenario. Besides, the following goals were pursued during the design of DiaFlux:

1. *Modularity*: To alleviate the reuse of (parts of) formalized knowledge, DiaFlux models are intended to be reused in different contexts. The modularization also helps to improve the maintainability of the knowledge base.
2. *Repetitive execution of subtasks*: Online monitoring involves the continuous observation of sensory data to detect fault states and initiate corrective action. Therefore particular actions need to be performed in an iterative manner.
3. *Parallelism*: Subtasks with no fixed order and dependency can be allocated to additionally spawned threads of control, and thus allow for their parallel execution. Expressing parallelism is especially necessary for mixed-initiative diagnosis, in which human and machine initiated examinations are carried out concurrently.
4. *Testability*: The evaluation of a knowledge base is an essential step prior to its productive use. We provide basic functionality for empirical testing and anomaly checks tailored to DiaFlux models.

Language Description

Wang et al. [10] studied several guideline representation models and identified common primitives for guideline creation. These were categorised as *actions*, *decisions* and for the representation of *patient state*. Actions denote specific clinical tasks like collecting data or clinical intervention. Decisions represent clinical decision making and guide the course of the care process. Patient states represent a patient's clinical status in the context of guideline application.

As DiaFlux models are based on flowcharts, we identified a minimal subset of node types to incorporate the necessary primitives to express a clinical protocol. To express the procedural aspect of the protocol, nodes can be connected by edges. There are no different types of edges, but they can be labeled with different types of conditions. The actual type of the condition depends on the type of node the edge starts at. They are used to evaluate the declarative knowledge with respect to the observed findings, e.g. the outcome of a given test or the status of a diagnosis. To obtain the semantics necessary for executability, we rely on an application ontology as an extension to the task ontology of diagnostic problem solving [11]. The application ontology defines the declarative knowledge consisting of findings and their ranges, solutions and treatments.

In the following, we enumerate and informally describe the different node types, before we give a toy example of a DiaFlux model in the introduced car fault diagnosis domain:

- **Start node:** A start node does not imply an action itself, but is a pseudo-node pointing to the node that represents the first action to take. Multiple start nodes can be modeled to provide distinct entry points into one DiaFlux protocol.
- **Test node:** Test nodes represent an action for carrying out a single test on activation of the node at runtime. This may trigger a question the user has to answer or data to be automatically obtained by sensors or from a database. Furthermore, the acquired information refines the knowledge about the patient state.
- **Solution node:** Solution nodes are used to set the evaluation of a solution, based on the observed findings.
- **Wait node:** Upon reaching a wait node, the execution of the protocol is suspended until the given time has passed.
- **Composed node:** DiaFlux models can be hierarchically structured as already defined ones can be reused as modules, represented by a composed node. This fulfills the aforementioned goal of modularity.
- **Exit node:** An exit node terminates the execution of a DiaFlux model and returns the control flow to the superordinate model. To express different results of a model, several distinct labeled exit nodes are supported.
- **Comment node:** For documentation of a protocol, comment nodes can be inserted at arbitrary positions. Though, they can be connected by edges and so be used to create semi-formal guidelines. They do not represent an action and are ignored during execution.

Figure 2 shows a DiaFlux module for handling the problem area "Battery", which has been established by another - superordinate - module (not shown in Figure 2). It is embedded into the wiki article containing further informal information about batteries.

The execution of the module starts at the *start node* "Start" (1), which is pointing to the node "Battery voltage" (2). It is a *test node*, which asks the user for the actual voltage of the battery. As "Battery voltage" is a question with a numerical range, the conditions that can be modeled on the outgoing edges, are checks against disjoint intervals. In case the battery's voltage is high enough to start the car (> 12.5V), the fault may be a connection problem to the electrical system. So, the user is asked whether the terminals are clean. In case they are, the problem has to be the automobile self starter or its cabling. Then, the execution reaches a *solution node* that evaluates the solution "Damaged Starter" as suspected (3). An attached comment node gives a hint for further elaboration, pointing out, that also the cabling may be the fault case.

Having rusty terminals, a connection problem is likely. The instructions of how to clean the terminals are a self-contained module, that can readily be reused in this protocol. Upon reaching the *composed node* "Clean Terminals" (4) the execution of the according protocol is started and "Battery Check" is stalled until the subordinate one is finished. In this example, the module "Clean terminals" exclusively consists of instructions to the user, so

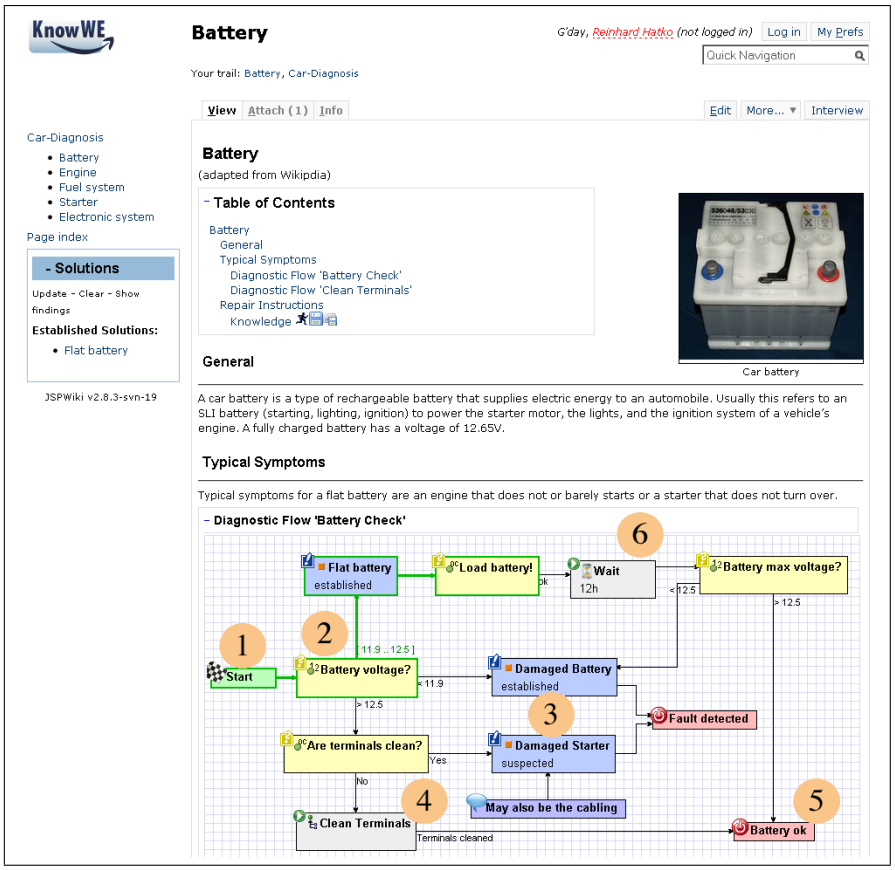

Figure 2: A guideline for diagnosing a car battery, embedded into a wiki article containing further information. The pathway of the current testing session is highlighted in green.

the only possible outcome of this procedure is "Terminals cleaned"(represented by an according exit node), which is the condition on the outgoing edge. After following the instructions and finishing the module, the *exit node* "Battery ok" (5) is activated, returning execution to the superordinate protocol (not shown in Figure 2).

In case the battery's voltage is too low to reliably start the engine (below 12.5V), different solutions can be established. If the voltage lies below 11.9V, then the battery is exhaustively discharged and considered broken. Therefore, the solution "Battery damaged" is established and the execution is ended by reaching the exit node "Fault detected". If the voltage is in the range between [11.9V, 12.5V], then it is probably too low for activating the self starter, but can be recharged. After establishing the solution "Flat Battery", the user is instructed to load the battery by the node "Load battery" (this basically is modeled by a one-choice question with the only one answer "ok"). Then the *wait node* (6) is activated suspending the execution for 12 hours. After this period has passed, the execution of the protocol is resumed. The user is asked for the actual battery voltage again, which is the battery's maximum voltage. If the voltage still is too low to start the car (< 12.5V), a "Damaged Battery" can again be established. If the voltage is sufficiently high, the taken exit node "Battery ok" indicates, that there must be another cause for the fault. After returning to the superordinate protocol, further steps can be taken to find the cause of the fault, e.g., a damaged engine. The outgoing edges in the superordinate protocol starting at the composed node representing "Battery Check" can decide how to proceed further, depending on the taken exit node.

Integration in KnowWE

We created an implementation of DiaFlux for the knowledge-based system d3web [12]. DiaFlux offers the possibility to model and execute protocols that employ declarative and inferential expressiveness provided by d3web.

An AJAX-based editor for DiaFlux is integrated into the Semantic Wiki KnowWE (cf. Figure 3), using its plugin mechanism [13]. The DiaFlux editor is on the one hand able to reuse ontological concepts that are readily available in the wiki's knowledge base. Those can simply be dragged into the flowchart. Depending on the type of object (finding, solution, DiaFlux model), a node of adequate type is created. On the other hand, the application ontology can be extended by creating new concepts from within the editor with an easy to use wizard. The model's source code is encoded in XML and integrated into the corresponding wiki article and saved and versioned together with it. This allows for further documentation of the protocol by tacit knowledge in the article. When the article is displayed in a web browser, the model visualization is rendered, instead of displaying its XML source code.

A related wiki environment for the collaborative creation of clinical guidelines is the Modelling Wiki (MoKi) [14], based on Semantic Media Wiki [15]. Originally it was designed for the creation of enterprise models using a visual editor, but it also has been used in the Oncocure project [16] to acquire clinical protocols for breast cancer treatment. Therefore, templates were defined within the wiki and later their content was exported into skeletal Asbru plans [17]. Though MoKi's visual editing capabilities for business pro-

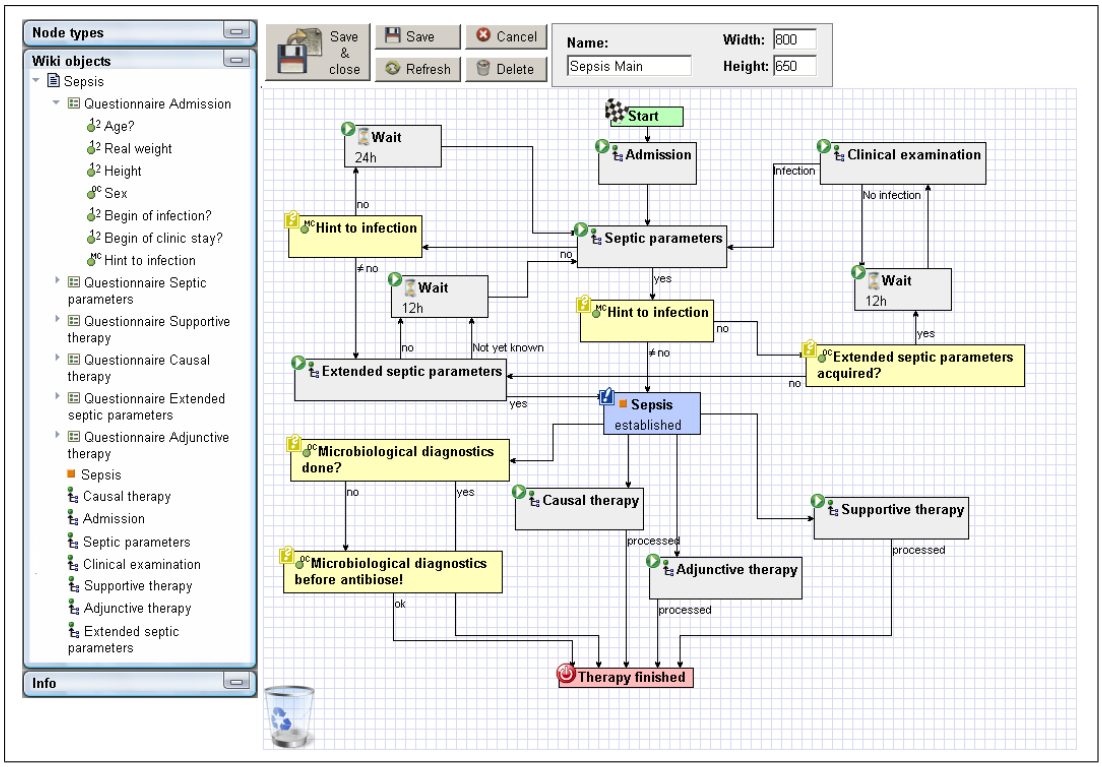

Figure 3: The main module of the sepsis diagnosis and treatment protocol, opened in the AJAX-based editor.

cesses, they were not employed to graphically model guidelines. Furthermore, the created Asbru plans are currently not executable within the wiki.

Development Process

For the development of DiaFlux models we propose the idea of the knowledge formalization continuum [2]: At first, informal information can be collected in wiki articles, e.g. about goals of a protocol. During the next step, a first semi-formal flowchart can be created using only comment, start and exit nodes and connecting edges. At this stage of formalization, the flowcharts can not be automatically executed, but "manually". For testing purposes the user can run through the flowchart by clicking on that outgoing edge of the active node, he wants to continue the pathway on. The taken pathway is highlighted for easier tracking. This especially is useful, when parallelism or hierarchically structured protocols are involved. The last step is the formalization into a DiaFlux model and the creation of the application ontology, resulting in a fully formalized and executable knowledge base. By following this process of gradual refinement, the entry barrier for domain specialists is quite low, while knowledge acquisition can start from the beginning.

Graphical modeling languages as a mediator to extract knowledge are also used in the *IT-Socket* of the research project *plugIT* [18]. Compared to our process of gradual refinement within the same modeling language, the IT-Socket employs semi-formal graphical models created by domains specialists, that are later formalized on a different level of abstraction.

Development Tools

Collaborative development requires to track the changes of all participants. Therefore, a frequent task is to compare different versions of a wiki article. For this purpose in general, wikis provide a textual diff comparing two versions of an article. As a diff of the XML source code is not very helpful for comparing a visual artifact like a flowchart, a more understandable diff is provided. On the one hand, a textual summary of the added, removed, and changed nodes and edges is generated. On the other hand, the previous and the current version of the DiaFlux model are shown next to each other, highlighting the changes in different colors for easy comparison, e.g. removed items are red in the previous version, added items are green in the current one, and changed items are highlighted in both versions.

After creating a knowledge base in KnowWE, a test session can directly be started from the wiki article containing it. Having used DiaFlux models, the current state of the protocol throughout the session can be observed. The traversed pathway through the flowchart is highlighted, in a similar manner as in the visual diff (cf. Figure 2). This immediate feedback considerably eases the interactive testing of the knowledge base.

4 Case Study

In the context of the project "CliWE" we used a prototype of the clinical wiki environment for the development of a protocol covering the diagnosis and therapy of sepsis. Sepsis is a syndrome of a systemic inflammation of the whole body. Despite the high mortality of this critical illness (30 to 60%) there are two main problems in sepsis therapy. First, it is essential to recognize that a patient fulfills sepsis criteria and second, if sepsis is diagnosed a complex medical therapy has to be initiated quickly. Today, so called patient data management systems are available in many intensive care units. With these systems, medical data are electronically available. In this context a clinical decision support system may be a reasonable solution for the above outlined practical problems, monitoring all patients for sepsis and support the physician in the initiation of sepsis treat-

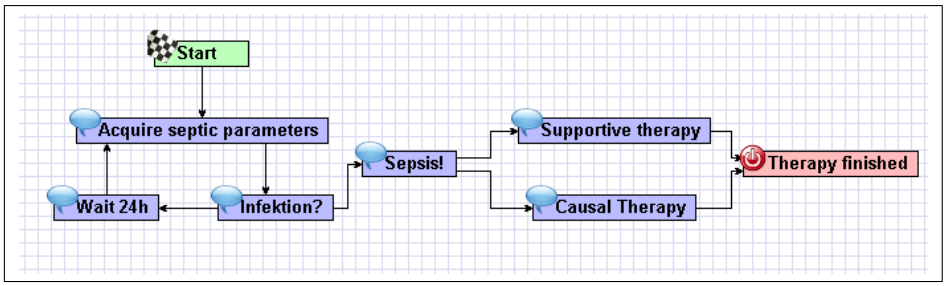

Figure 4: An early semi-formal version of the sepsis protocol.

ment.

The knowledge base was developed in accordance to the official guideline by the German Sepsis Society [19]. It is a textual guideline of about 80 pages describing the prevention, diagnosis, and therapy of sepsis. Our formalization of the guideline contains so far the diagnostics and parts of the therapy together with some common tasks for patient admission (cf. Figure 3). At the moment it contains about 50 nodes in eight modules with several possible pathways, depending on how the diagnosis can exactly be established and the course of the therapy. The upper part of the main DiaFlux model contains knowledge about the decision making and the lower part contains knowledge about the treatment.

The diagnosis task involves the assessment of up to eight clinical parameters (conducted in the modules "Septic parameters" and "Extended septic parameters") and an established or suspected infection. The monitoring is repeated until a sepsis can be established within different cycles depending on which parameters are acquired and their evaluation. If there is enough evidence to support a suspected sepsis, then a warning to the clinician is generated. If the clinician agrees with the conclusion, the diagnosis "Sepsis" is established and instructions for starting the therapy are given. The treatment for sepsis consists of the three bundles *causal therapy* (treating the cause of the infection), *supportive therapy* (stabilizing the patients circulation) and *adjunctive therapy* (supporting fighting off the infection). Those bundles are modeled as self-contained modules and reused as composed nodes in the main module.

Experiences

The knowledge acquisition mainly took place in two workshops, approximately six hours each, involving two domain experts. The DiaFlux editor was handled by a knowledge engineer, entering the knowledge artifacts provided by the domain specialists. The remaining participants followed the authoring process on a projector.

During the first session we followed the idea of the knowledge formalization continuum and started with textual descriptions of most modules. As a second step, we created semi-formal flowcharts giving an outline of the protocol, as exemplified in Figure 4. Next, we started to further formalize these flowcharts into executable DiaFlux models and to create the according declarative knowledge. The second session began with the acquisition of test cases of typical sepsis patients. As they were only informally entered in a wiki article and not executable so far, we stepped manually through the model by highlighting the correct pathway. The found inconsistencies were corrected during the second half of the session, together with further elaboration of the knowledge base. In a third session of about one hour, one of the experts created a small module by himself, while being observed by a knowledge engineer. The expert shared his screen using an internet screen sharing software and was supported in formalizing the knowledge and the usage of the DiaFlux editor.

Overall, the wiki-based approach showed its applicability and usefulness, as the combination of formal and informal knowledge and its gradual refinement was intensely used during the acquisition of the protocol and the test cases. Further, the developed knowledge base was accessible to all participants immediately after the workshops, as it took place in a password protected wiki, which can be accessed over the internet.

So far, the knowledge acquisition was conducted in workshops involving domain experts and knowledge engineers. After the initial workshops and the successful teleknowledge acquisition session, we are confident to proceed with further workshops, that require minimal support by the knowledge engineers.

5 Conclusion

This paper presented work in the context of the project "CliWE - Clinical Wiki Environments" for collaborative development and evolution of clinical decision-support systems. We introduced a language that can incorporate declarative and procedural diagnostic knowledge for modeling executable clinical protocols. Its main focus is simplicity for the usage by domain specialists. DiaFlux is integrated into the Semantic Wiki KnowWE to support the collaborative development by a community of experts. The case study demonstrated the applicability and benefits of the approach during the development of a clinical protocol for sepsis diagnosis and treatment. Due to the wiki-based approach the knowledge can evolve easily. It is accessible without depending on specialized software, as long as an internet connection is available. Furthermore, domain specialists can almost instantly start contributing. Formalization of the knowledge can then happen at a later time, after familiarizing with the semantics.

As next steps we plan the integration of refactoring capabilities into the editor, for the easier evolution of DiaFlux models. We will also enhance the tool support for the gradual formalization. As there rarely is only one single opinion in medicine, we will support different "medical schools" represented by the contributing experts by the possibility to engineer *variants* of DiaFlux models. In the future, we are planning the collaborative development by a community of experts, connected by KnowWE.

References

[1] Schaffert, S., Bry, F., Baumeister, J., Kiesel, M.: Semantic wikis. IEEE Software **25**(4) (2008) 8–11

[2] Baumeister, J., Reutelshoefer, J., Puppe, F.: Continuous knowledge engineering with semantic wikis. In: CMS'09: Proceedings of 7th Conference on Computer Methods and Systems (Knowledge Engineering and Intelligent Systems). (2009) 163–168

[3] Buffa, M., Gandon, F., Ereteo, G., Sander, P., Faron, C.: SweetWiki: A semantic wiki. Web Semantics **8**(1) (2008) 84–97

[4] Baumeister, J., Reutelshoefer, J., Puppe, F.: Markups for knowledge wikis. In: SAAKM'07: Proceedings of the Semantic Authoring, Annotation and Knowledge Markup Workshop, Whistler, Canada (2007) 7–14

[5] Hommersom, A., Groot, P., Lucas, P., Marcos, M., Martínez-Salvador, B.: A constraint-based approach to medical guidelines and protocols. In Teije, A.t., Miksch, S., Lucas, P., eds.: Computer-based Medical Guidelines and Protocols: A Primer and Current Trends. Volume 139 of Studies in Health Technology and Informatics. IOS Press (2008) 213–222

[6] Boxwala, A.A., Peleg, M., Tu, S., Ogunyemi, O., Zeng, Q.T., Wang, D., Patel, V.L., Greenes, R.A., Shortliffe, E.H.: GLIF3: a representation format for sharable computer-interpretable clinical practice guidelines. J. of Biomedical Informatics **37**(3) (2004) 147–161

[7] Fox, J., Johns, N., Rahmanzadeh, A.: Disseminating medical knowledge: the proforma approach. Artificial Intelligence in Medicine **14**(1-2) (1998) 157 – 182 Selected Papers from AIME '97.

[8] de Clercq, P., Kaiser, K., Hasman, A.: Computer-interpretable guideline formalisms. In ten Teije, A., Miksch, S., Lucas, P., eds.: Computer-based Medical Guidelines and Protocols: A Primer and Current Trends. IOS Press, Amsterdam, The Netherlands (2008) 22–43

[9] Peleg, M., Tu, S., Bury, J., Ciccarese, P., Fox, J., Greenes, R.A., Miksch, S., Quaglini, S., Seyfang, A., Shortliffe, E.H., Stefanelli, M., et al.: Comparing computer-interpretable guideline models: A case-study approach. JAMIA **10** (2003) 2003

[10] Wang, D., Peleg, M., Tu, S., Boxwala, A., Greenes, R., Patel, V., Shortliffe, E.: Representation primitives, process models and patient data in computer-interpretable clinical practice guidelines:: A literature review of guideline representation models. International Journal of Medical Informatics **68**(1-3) (2002) 59 – 70

[11] Baumeister, J., Reutelshoefer, J., Puppe, F.: KnowWE: A semantic wiki for knowledge engineering. Applied Intelligence (2010)

[12] Baumeister, J., et al.: The knowledge modeling environment d3web.KnowME. open-source at: http://d3web.sourceforge.net (2008)

[13] Reutelshoefer, J., Lemmerich, F., Haupt, F., Baumeister, J.: An extensible semantic wiki architecture. In: SemWiki'09: Fourth Workshop on Semantic Wikis – The Semantic Wiki Web (CEUR proceedings 464). (2009)

[14] Ghidini, C., Kump, B., Lindstaedt, S.N., Mahbub, N., Pammer, V., Rospocher, M., Serafini, L.: MoKi: The enterprise modelling wiki. In: ESWC'09: The Semantic Web: Research and Applications. Volume 5554 of LNCS., Springer (2009) 831–835

[15] Krötzsch, M., Vrandecić, D., Völkel, M.: Semantic MediaWiki. In: ISWC'06: Proceedings of the 5th International Semantic Web Conference, LNAI 4273, Berlin, Springer (2006) 935–942

[16] Eccher, C., Rospocher, M., Seyfang, A., Ferro, A., Miksch, S.: Modeling clinical protocols using Semantic MediaWiki: the case of the oncocure project. In: K4HelP: ECAI 2008 Workshop on the Knowledge Management for Healthcare Processes, University of Patras (2008) 20–24

[17] Miksch, S., Shahar, Y., Johnson, P.: Asbru: A task-specific, intention-based, and time-oriented language for representing skeletal plans. In: UK, Open University. (1997) 9–1

[18] Woitsch, R., Utz, W.: The IT-Socket: Model-based business and IT alignment. In Weghorn, H., Isaías, P.T., eds.: IADIS AC (1), IADIS Press (2009) 141–148

[19] German Sepsis-Society: Sepsis guideline. http://www.sepsis-gesellschaft.de/DSG/Englisch