

# Capture and Refactoring in Knowledge Wikis

## Coping with the Knowledge Soup

Joachim Baumeister<sup>1</sup>, Jochen Reutelshoefer<sup>1</sup>,  
Fabian Haupt<sup>1</sup> and Karin Nadrowski<sup>2</sup>

1) Institute of Computer Science, University of Würzburg, Germany

2) Institute of Animal Ecology, University of Giessen, Germany

Today's scientific research is mostly acquired and shared using internet resources. In this context web collaboration has become more important these days. The paper presents the semantic knowledge wiki KnowWE that is used to capture and share ontological knowledge together with explicit problem solving knowledge in an open web environment. We also sketch a scientific research project using KnowWE as a knowledge pool and we show that distributed knowledge formalization poses new research questions concerning the quality and refactoring of knowledge.

## 1 Introduction

The Web is a significant infrastructure for today's research, especially for communicating and distributing new research results. Similarly, web-collaboration has become more important for collecting and sharing scientific knowledge with a wide range of users.

Starting in early years with static web pages presenting new research results and mailing lists to communicate and discuss on particular topics, today's researchers use more dynamic and interactive systems. In this paper we introduce the system KnowWE that enables researchers to effectively and interactively elicitate and share knowledge. KnowWE [2, 12] is a semantic knowledge wiki that extends a standard wiki system by the possibility to explicitly represent ontological concepts and problem-solving knowledge. In the context of a scientific community a semantic wiki can be used to:

- Organize a specific scientific topic in subsections and weave the related concepts by (semantically) annotated links.

- Publish relevant information of subsections in the form of text and multimedia on the web in a simple and flexible manner.
- Annotate textual/multimedia content with ontological concepts describing the domain in order to allow an explicit linkage of the subsection's contents. More importantly, semantic annotation allows semantic search and semantic navigation.

As a semantic knowledge wiki we extend “standard” semantic features of semantic wikis by explicit problem solving knowledge, thus we are additionally able to

- Attach problem solving knowledge: Whereas a semantic wiki describes concepts and properties between concepts, we can additionally define more expressive knowledge like rules and models, that are able to derive the instance of an existing concept.

In the following we give a brief introduction to the system KnowWE showing its content/knowledge acquisition component as well as its use for knowledge sharing.

## The Semantic Knowledge Wiki KnowWE

In most semantic wikis every concept is described by a distinct wiki page, where the concept is described in a textual manner including multimedia content for the concept. Text phrases are semantically annotated with properties of a given wiki ontology; in most cases new properties can be defined in an ad-hoc way. Recent examples of semantic wikis are Semantic MediaWiki [9], IkeWiki [13], and SWIM [10]. The semantic knowledge wiki KnowWE [12] additionally allows the intuitive capture and use of explicit problem-solving knowledge that is applied to derive particular concepts.

While standard semantic wikis contain concepts and properties between concepts, we are further able to define external problem-solving knowledge like rules/models to derive instances of particular concepts. Knowledge in semantic wikis is often used in order to state semantic queries uncovering specific concepts or to facilitate semantic navigation. In semantic knowledge wikis the knowledge is typically used to derive an appropriate concept as a *solution* for facts entered by a user. These *facts* are interpreted as instances of wiki concepts, but they are not globally stored in the wiki, since they are only present in the actual user-session as findings of the user's case. The user of a semantic knowledge wiki typically browses the contents of the wiki in a classic way by following (semantic) links within the web page or by using (semantic) search forms. In addition, an interactive interview can be initiated in order to derive a suitable solution for the stated problem description in a more knowledge-based style. Users can also click on *inline answers*, i.e., clickable text phrases, in order to enter findings into the problem-solving session. Inline answers are generated on a basis of semantic knowledge annotations made in the particular wiki articles. In order to show the basic elements of the semantic knowledge wiki we describe its use and its edit mode. Figure 1 shows a sports advisor wiki, which is an example system for demonstrating the functionality of the system. Here, the sports form *Swimming* is shown by a description of the sports form, a picture and interactive

**Swimming** G'day, [JochenReuetelshofer](#) (authenticated) [Log out](#) [My Prefs](#)

Quick Navigation

Your trail: [Swimming](#), [Tennis](#), [Swimming](#), [SportsAdvisor](#), [Running](#), [SportsAdvisor](#), [Soccer](#), [Golf](#), [Swimming](#), [Golf](#)

**View** **Attach (1)** **Info** **Edit** **More...**

Some parts are taken from Wikipedia, the free encyclopedia ([Wikipedia article](#))

Swimming is movement by humans or animals [in water](#), usually without artificial assistance. Swimming is an activity that is both useful and [recreational](#) for many species. Its primary uses are bathing, cooling, travel, fishing and escape. An individual's ability to swim can be judged by speed or stamina. Animals with lungs have an easier time floating than those without. Almost all mammals can swim by instinct. Bats, kangaroos, moles and sloths can swim, despite their rather strange shapes. The few exceptions include apes and possibly giraffes. Land birds can swim or float for at least some time. Ostriches, cassowaries and tortoises can swim.

- Water [sport](#)
- Recommended [Age](#): older than 4 years
- Competitive [sport](#): no
- [Flexibility](#): spontaneous
- Not a team [sport](#)
- [Motivation](#): endurance, staying fit, losing weight
- [Medical](#) [risks](#): vascular diseases, general skin problems
- Low Risk
- [Trained](#) [person](#): yes
- Medium [difficulty](#)
- [Running](#) [related](#): no
- [Duration](#): indefinite

Motivation

meet new people

having fun

reducing stress

endurance

staying fit

losing weight

**Established Solutions:**

- [Swimming](#) [1.0;0.11]

**Suggested Solutions:**

- [Running](#) [0.5;0.2]
- [Tennis](#) [0.5;0.25]

Figure 1: Providing new facts given as inline answers in the knowledge wiki KnowWE.

elements within the text. The user can, for example, enter new facts by clicking on links in the system; in the shown example the user enters some values for the question *Motivation*. When new facts are entered, the system instantly derives solutions that can be inferred based on the given findings. All appropriate solutions are shown in the right pane of the wiki, for example the solution *Swimming* was actually derived as a suitable solution, but the solutions *Running* and *Dancing* are additionally suggested for further consideration. By clicking on the solution names the user can easily navigate to the corresponding wiki articles describing the sport forms in more detail. Moreover, the system can explain the derivation of results by clicking the solution name.

Derivation knowledge for every solution is entered together with the remaining content of the wiki article. By clicking the edit button the wiki page can be modified. Here, the user can insert a special knowledge topic (Kopic) into the standard text but is also able to semantically annotate particular text phrases with concepts of the application ontology (including solutions and findings of the shown sports advisor example). Existing annotations are used for inline answers in the view mode of the wiki. Figure 2 shows

the corresponding edit page of the wiki article. Here, the usual content of the article is entered together with explicit problem-solving knowledge. In this case (scoring) rules are used to exclude the solution *Swimming* based on findings entered by the user. Further knowledge annotation methods and markups are discussed in the next section.

**Swimming** G'day, [JochenReuetelshöfer](#) (authenticated) [Log out](#) [My Prefs](#)

Quick Navigation

Your trail: [Swimming](#), [TiRexConfig](#)

[Edit](#) [Attach \(1\)](#) [Info](#) [Help](#) [More...](#)

[Save](#) [Syntax Check](#) [Preview](#) [Cancel](#) [Redo](#) [Undo](#)

Swimming is movement by humans or animals [in water  $\Leftrightarrow$  explains:: Medium of sport = water], usually without artificial assistance. Swimming is an activity that is both useful and [recreational  $\Leftrightarrow$  asks:: Motivation] for many species. Its primary uses are bathing, cooling, travel, fishing and escape. An individual's ability to swim can be judged by speed or stamina. Animals with lungs have an easier time floating than those without. Almost all mammals can swim by instinct. Bats, kangaroos, moles and sloths can swim, despite their rather strange shapes. The few exceptions include apes and possibly giraffes. Land birds can swim or float for at least some time. Ostriches, cassowaries and tortoises can swim.

```
<Kopic terminology="SportFindings..SportFindings_KB">
<Comment-section>Some facts describing the form of sport
'Swimming'.</Comment-section>
<Rules-section>
  IF Medical restrictions = skin allergy    OR    Medical restrictions = general
  skin problems
  THEN Swimming = N7
</Rules-section>
```

[link](#) [h1](#) [h2](#) [h3](#) [hr](#) [br](#) [pre](#) [dl](#) [bold](#) [italic](#) [mono](#) [sup](#) [sub](#) [strike](#) [toc](#) [tab](#) [table](#) [img](#) [Edit Assist](#)

[code](#) [quote](#) [sign](#)

Tab Completion (keyword+Tab)  Smart Typing Pairs

Figure 2: Edit mode of the wiki: Text and multimedia is entered together with explicit knowledge, here derivation rules for the solution *Swimming*.

The rest of the paper is organized as follows: In Section 2 we describe alternative markups for knowledge elicitation in the semantic knowledge wiki KnowWE; we shortly describe the utilized upper knowledge ontology and introduce the novel knowledge representation eXtensible covering lists to be used during the evolutionary formulation of models, and we show how text phrases can be annotated semantically by semantic knowledge annotations or implicitly by using structured texts. Both annotation variants can replace the explicit definition of covering knowledge. A current project is described in Section 3 where we focus on the recent projects we are facing during the evolutionary development process. We describe how to detect “knowledge soup” and we sketch methods to reorganize the knowledge wiki structure by refactoring methods. Some conclusions of the presented work are given in Section 4

## 2 Knowledge Acquisition Interfaces

In this section, we explain several possibilities to enter the problem-solving knowledge in the text. As we recommend the use of simple knowledge structures for non expert users, we offer alternative ways for defining simple relations between findings and solutions. For more experienced users we provide the versatile *Kopic* tag for the definition of complex knowledge in different ways. First we introduce an Upper Ontology to explain the conceptual organization of knowledge within the system.

### 2.1 The Upper Ontology for Semantic Knowledge Wikis

Independently of the utilized input methods, all formalized knowledge can be traced back to the proposed upper ontology depicted in Figure 3a). We are giving a short overview of the basic ontology structure in this section. Although the figure gives only an overview of the ontology, as some of the details are left out for clarity, one can see the hierarchical concept interaction. The world state can be described by the key concept *Input*. Represented by a set of variables the overall set of inputs holds the world state of the knowledgebase. Groupings of inputs are put together through *Questionnaires* collecting them in meaningful clusters. The primary subclasses of input are *InputChoice* and *InputNum* which capture discrete (named) values and numerical ranges respectively. The corresponding value subclassing of *Value* is assigned to each input class. The special subclass *Solution* of *InputOneChoice* is used to represent an input made by the knowledge base. A solution therefore depicts the final output of a problem-solving session with its derived knowledge. As such, the solution has its value range limited to one of *Derived*, *Suggested*, *Undefined* or *Excluded*, representing the respective outcome of a problem solving process.

Any particular case is modeled as an instance of the *PSSession* class where the user inputs and therefore the inherent case knowledge is hold. Problem-solving sessions are created separately for each user session to guarantee independent user sessions, regardless of the number of users. The sessions themselves are structured by the *Finding* concept, glueing together input and value instances representing answered questions.

To allow a very general usage of the upper ontology, the derivation knowledge is modeled in an abstract manner. The key concept in this part of the ontology is *LogicExpression*. *Literal* and *CompositeExpression* subclass *LogicExpression*. The latter paves the way for the composition of logical expressions over findings using the subclasses *Conjunction*, *Disjonction*, or *Negation*. The properties *explains* and *isContradictedBy* connect this full logical expressiveness to the solution concept. These properties represent the different problem solving methods one can employ. This allows for an easy extension of the upper ontology for new problem solving methods.

### 2.2 eXtensible Covering Lists (XCL)

To support collaborative development of formal knowledge systems we introduce a knowledge representation - eXtensible Coverings Lists - which is an extension of set-covering

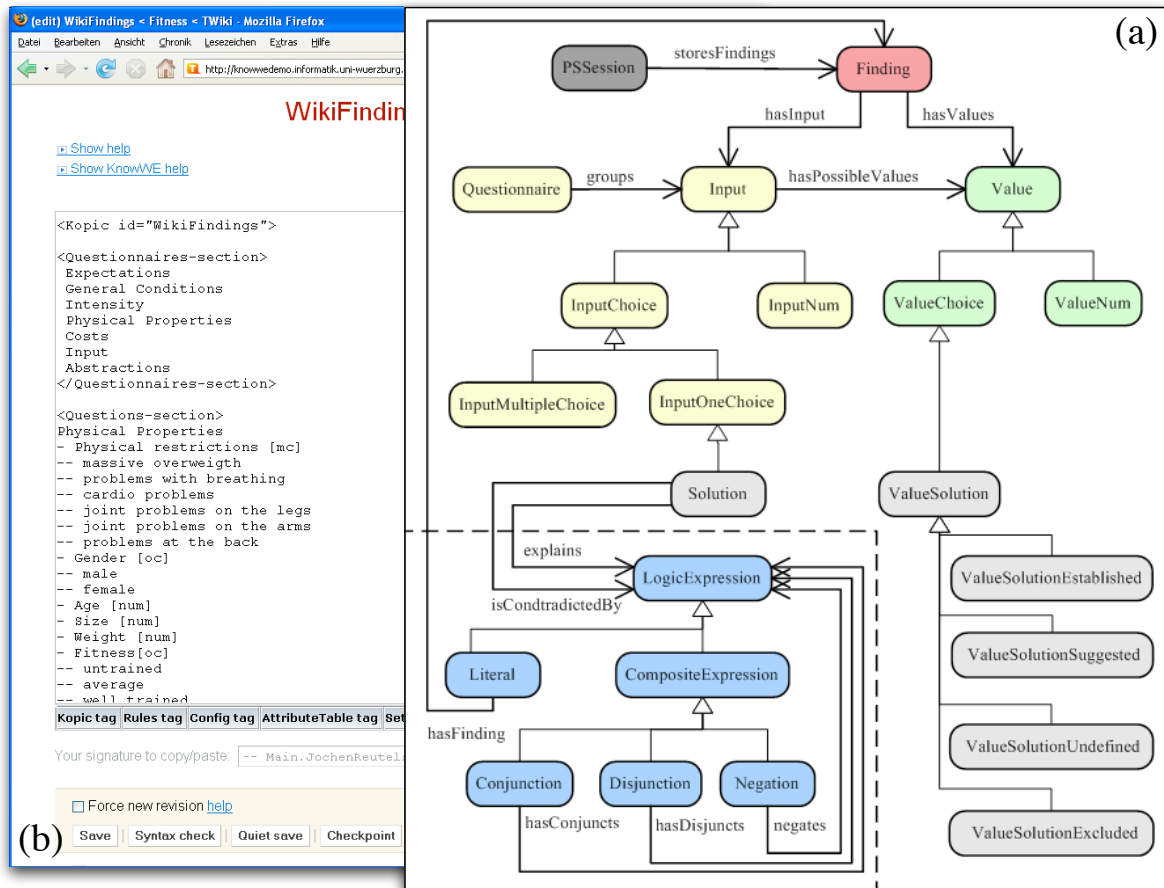


Figure 3: a) The upper ontology of the semantic knowledge wiki KnowWE. b) a part of the input definitions (WikiFindings page) of the sports advisor demo.

models [11]. The XCL representation is designed with two goals in mind: Simplicity of use and the support for an evolutionary development process foreseeing the incremental extension of an initially simple model in small steps. A simple XCL model is basically a list of findings that typically occur when a particular solution is present. In the following example an XCL model for the solution *Swimming* is shown. Each line represents a positive coverage of the finding by the solution and is called *explains* relation. The order of the listed findings is not relevant for the inference process and is thus arbitrary. If the domain knowledge is already available as informal text, then it denotes a simple task to transfer the key findings described in the text into basic findings contained in an XCL.

---

```
1 Swimming {
2   medium = water,
3   Type of sport = individual,
4   Training goals ALL {endurance, stress reduction},
5   Running costs = medium,
6   Trained muscles = upper part,
7   Trained muscles = back
8 }
```

---

During the inference process the best rated solution is chosen. A solution is rated by comparing the findings defined in the XCL to the findings entered by the user. The rating of a solution is expressed by its *covering score*. This numeric score is mapped to four predefined solution states: *Unclear* (default), *Suggested*, *Established* and *Excluded*.

The basic XCL representation can be extended in multiple ways: Besides the simple listing of findings shown above, the XCL representation offers further elements to extend/refine the expressiveness and selectivity of the covering model that are briefly discussed in the following (the textual markup is given in parentheses):

**Exclusion knowledge** [--]: A fulfilled relation which is marked as an exclusion makes the derivation of the solution impossible. This type of relation is called *isContradictedBy* and it sets the state of the solution to *Excluded* when fulfilled. Such a constraining relation is defined by two minus signs in brackets ( [-- ]) at the end of the relation line, as shown in line 7 of the markup shown below.

**Required relations** [!]: Relations can be marked as *required* by using a bracket containing an exclamation mark ([!]). Then a solution can only be established as a possible derivation if all required relations are fulfilled, i.e., the corresponding findings were positively entered by the user. An example is shown in line 2 in the following markup.

**Sufficient relations** [++]: By adding a bracket with two plus signs ([++]) to a relation we define this solution to be a sufficient relation. Then, the solution is always established if the corresponding finding is fulfilled. We call this relation *isSufficientlyDerivedBy*. It is important to know that contradicting relations are dominating sufficient relations.

**Adding weights** [ *num*]: In the initial version every relation is equally important when compared to the other relations of the XCL, thus having the default weight 1. The default weight can be overridden for relations in order to express their particular importance. In the textual notation the weight is then entered in brackets at the end of the relation definition, for example [2] to double the weight of a relation. See line 8 in the following markup for an example.

**Logical operators** (AND, OR): A complex relation can be created by combining relations by logical operators. For example in line 2 of the model shown below the

findings *medium = water* and *Type of sport = individual* are connected by the logical or-operator (OR). The resulting knowledge demands that either *medium = water* or *Type of sport = individual* needs to be observed to fulfill the relation. The three basic operators of propositional logic *or*, *and* and *not* can be used.

**Threshold values :** When rating a solution the numeric covering score is mapped to a solution state. The mapping function is defined by the threshold values (**establishedThreshold** and **suggestedThreshold**). In most cases the internal default threshold values are adequate, but for distinct solutions they can be overridden as shown in line 12-13 of the following example model. In the example, 70% of the expected and observed findings need to be correctly observed to set the solution to the state *Established*, and 50% to set the solution to the state *Suggested* (higher states overwrite lower states). Further, with **minSupport** we specify the fraction of findings defined in the XCL model that need to be set by the user in order to activate the solution's rating process.

The following markup shows a refined version of the previous model of the solution *Swimming* using the described elements.

---

```
1 Swimming {
2   medium = water OR Type of sport = individual [!],
3   My favorite sports form = swimming [++],
4   Training goals ALL {endurance, stress reduction},
5   Favorite color IN {red, green, blue},
6   Running costs = medium,
7   Running costs = nothing [--],
8   Trained muscles = upper part [2],
9   Trained muscles = back [2],
10  Physical problems = skin allergy [--],
11  Type of sport = group [--],
12 }[ establishedThreshold = 0.7,
13   suggestedThreshold = 0.5,
14   minSupport = 0.5
15 ]
```

---

Essentially, some of the already defined relations were refined by relational extensions like sufficient, contradicting and necessary properties.

## 2.3 Semantic Knowledge Annotations

KnowWE offers semantic annotations of text phrases as already known by other Semantic Wiki systems like for example Semantic MediaWiki [9]. Text parts can be tagged with the relations introduced above in Section 2.2, for example by **explains**,

`isContradictedBy` and `isSufficientlyDerivedBy`. The following example shows sentences describing the solution swimming, where text phrases are annotated by `explains` and `isContradictedBy` relations. For example, in lines 1–2 a relation between the solution *Swimming* (the concept described by the page) and the finding *Medium = in water* created: the first expression after the opening brace and before the “`<=>`” is the textual part of the sentence that will be rendered in the view mode of the article, shown for the finding *Motivation* in Figure 1. The following part of the annotation states the name of the property (e.g. `explains`, `isContradictedBy`) followed by two colons. After that, the actual finding related to the solution concept is specified, i.e., the range of the given property. In the topic view the annotated text can be used as an interview method with inline answers as shown for example in Figure 1.

---

```
Swimming is the most common form of [water sports <=> explains::
Medium = in water]. Swimming is good for successfully [reducing
stress or to train endurance <=> explains:: Training Goals =
stress alleviation OR Training Goals = endurance]. It only
should be avoided when [cardio problems <=> isContradictedBy::
Physical restrictions = cardio problems] are present. Further,
Swimming is quite inexpensive [explains:: Running Costs = low].
```

---

The semantic annotation of existing sentences means reduced knowledge acquisition workload compared to the explicit markups introduced before. Although standard wiki text is tightly integrated with formal knowledge, the readability of the text suffers from annotations as shown above. For this reason, we introduce *structured texts* where non-annotated text phrases can automatically be parsed for knowledge annotations.

## 2.4 Structured Texts for Ad-Hoc Knowledge Formalization

NLP techniques can be used for annotating distinct parts of the wiki text internally. In the context of our work, we are able to use “structured texts” since 1) the available text can be mapped to a rather simple knowledge representation (covering relations), and 2) we can employ the given application ontology as background knowledge for the concept identification task. To generate the formal knowledge, the natural language expressions are matched to the concepts of the application ontology. Using this technique we assume that a distinct block of a wiki article is tagged as a structured text. Then, this block is parsed in order to identify findings which are compiled to covering relations (implicitly annotated by `explains`). In a first step we are working with semi-structured texts such as bullet lists. In the following example in Figure 4a we show a bullet list in standard wiki syntax, where each line contains one (combined) finding explaining the solution *Swimming*. While the inline annotations expect exact matches, we applied some lightweight linguistic methods for matching findings in structured texts. For example simple string matching combined with stemming and synonym lists already yield fairly good results. In the simplest case we can identify an input name that is defined in the

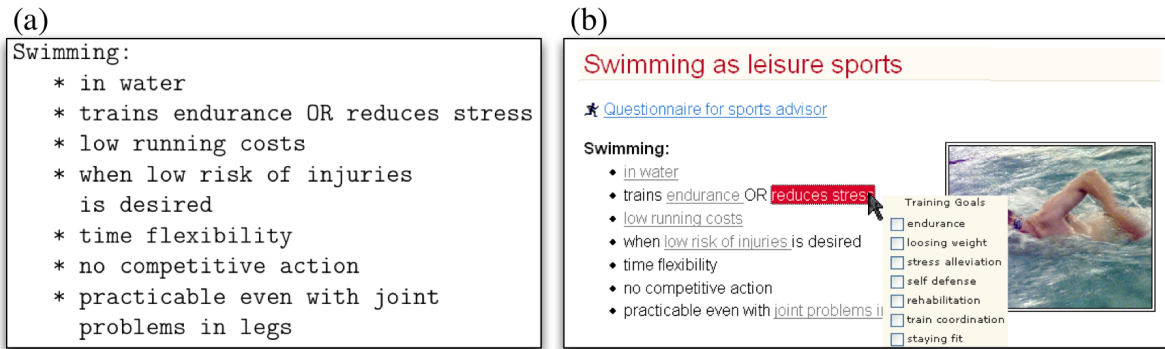


Figure 4: a) Wiki syntax of a bullet list in structured text, b) automatically annotated text phrases in a semi-structured text.

application ontology together with a corresponding value name, for example as found in line 5 of Figure 4a: the text phrase “when low risk of injuries is desired” yields the finding “risk of injury = low”. The finding defined by this input-value-pair tuple can be added as a covering relation of the solution *Swimming*. All identified findings are implicitly annotated and can be used to provide inline answers as shown in Figure 4b.

## 2.5 Further knowledge representations using the Kopic tag

Although we experienced the presented XCL models to be an expressive and intuitive knowledge representation, we additionally provide a wider range of other knowledge representations like (scoring) rules and (heuristic) decision trees. Those are recommendable for more experienced users that want to express complex relations between inputs and solutions. The example in Figure 2 shows the definition of a Kopic-tag defining rules for the solution *Swimming*. Various forms of knowledge representations for KnowWE are introduced in [4].

## 3 Current Experiences from a Case Study: BIOLOG Wissen

The project *BIOLOG Wissen* (BIOLOG Knowledge) was formerly known as LaDy (landscape diversity) and was introduced in [3]. BIOLOG Wissen serves as a web-based application for the collaborative construction and use of a decision support system for landscape diversity. It aims to integrate knowledge on causal dependencies of stakeholders, relevant statistical data, and multimedia content. Figure 5 shows the main screen of the system with some solutions already derived. The participating domain specialists have neither background in knowledge representation nor in ontology engineering, and therefore the interfaces need to be as simple and intuitive as possible.

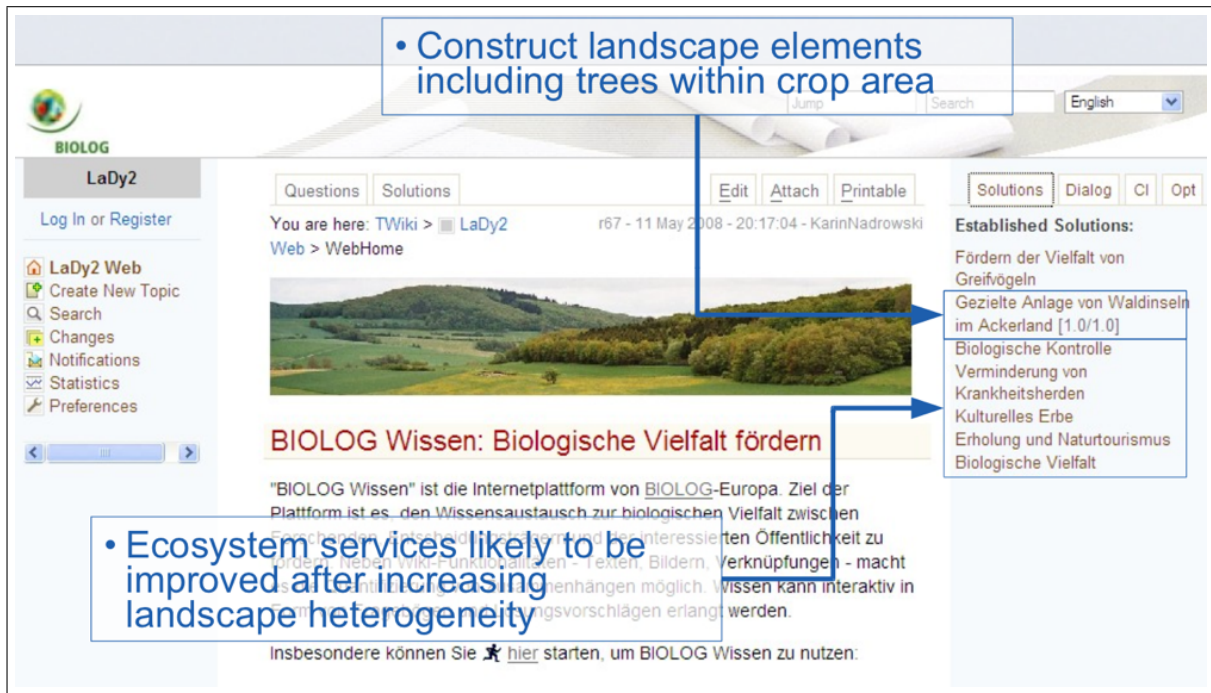


Figure 5: The main screen of BIOLOG Wissen with some already derived solutions.

### 3.1 Distributed Knowledge Elicitation

During this project we followed the evolutionary collaborative development cycle *SER* as proposed by Fisher [7]. The process is depicted in Figure 6 and basically consists of the three phases *seeding* (building some useful examples by a wiki champion<sup>1</sup>), *evolutionary growth* (let all people participating to extend the system) and *re-seeding* (a refactoring of the existing contents including restructuring, alignment and evaluation of the knowledge). In the seeding phase the wiki champion added basic structure and meaningful content as examples to clarify the system. Then, the evolutionary growth phase was initiated by inviting additional authors to contribute to the wiki. In the various kick-of meetings we learned that a simple covering list representation was experienced to be intuitive and suitable in this phase, since almost no domain specialists were trained in knowledge representation and reasoning. The simple representation and inference algorithm was quickly adopted and the users were able to implement simple models of the targeted sub-domains. After these simple examples the requirements of the users concerning the expressivity of the knowledge representation grew. In many cases these requirements could be covered by extended covering lists introduced in the previous section. In a few cases we thought about transforming the initial knowledge model to a rule base, where knowledge can be expressed in an even more flexible way.

In summary, the knowledge wiki offered a comprehensive platform for both exchanging textual as well as multimedia content of the domain and for implementing explicit knowl-

<sup>1</sup>A *wiki champion* is a selected person responsible for the structure and quality of the overall wiki. He/she guides new users and encourages the extension/creation of the particular domains.

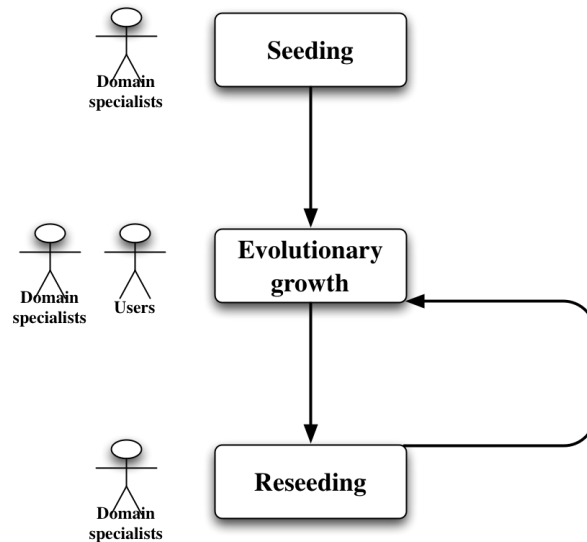


Figure 6: The seeding, evolutionary growth and re-seeding development cycle proposed by Fisher [7].

edge on ecosystem behaviour. Thus Biolog Wissen provides a service to condense and to communicate knowledge needed for an efficient management of ecosystem services.

### 3.2 The Creation of the Knowledge Soup

In the phase *evolutionary growth* new pages, concepts, and knowledge were added in a rather unorganized manner. As a consequence the system arrived at a state that can be called a “knowledge soup”, i.e., a large collection of intermixed elements of text, knowledge, and links that is hardly usable in a coherent manner. Therefore the system’s utility decreases drastically and a re-seeding phase becomes necessary.

John Sowa [14] defined knowledge soup as “fluid, loosely organized, dynamically changing contents of the human mind” and described examples for knowledge soup such as *overgeneralization*, *abnormal conditions*, and *incomplete definitions* that can be found in knowledge. Transferred to the situation of knowledge engineering in wikis we can identify knowledge soup as an unstructured collection of knowledge that is loosely organized in an informal manner.

Using a structured process, the creation of knowledge soup does not necessarily happen, for example, if the organization of the system and the concepts to be used are predefined by the wiki champion in a very careful way. However, in little understood and uncertain domains it is very unlikely that the structure of the entire domain can be modeled in advance. Furthermore, developers often feel more motivated and effective if they are able to extend and modify the given structure by the needs of the actual artifact that they are trying to model. For this reason it is very likely that evolutionary modeled knowledge becomes “messy” at some point and that a reorganization becomes necessary in a subsequent step. In the context of our own projects with knowledge wikis

we often observed this gradual decrease of knowledge structure during the development phase. With the initial experiences we have made so far we developed a set of *detectors of knowledge soup* that try to measure the structure of the system in a rather qualitative way.

### **Detectors of Knowledge Soup**

1. No uniform use of the application ontology
2. No uniform size of the knowledge bases recently added to the system; in this case, we especially focus on small knowledge bases
3. New articles are organized in a quite flat/non hierarchical structure

It is easy to see that identifying the described detectors needs a thorough analysis of the wiki content. Visualization methods can help the developer in an effective way. In the following we propose visualization methods for analyzing the use of the application ontology and we discuss some useful refactoring methods and the context in which they can be put to use.

#### **3.2.1 Uniform Use of Application Ontology**

This detector tries to identify whether the concepts of the application ontology are used in a regular way, for example whether the new articles and their knowledge, respectively, have introduced many new concepts and only reused very few concepts of the application ontology. A disappropriate frequent use of new concepts mostly indicates that the existing application ontology should be extended by some new concepts and/or newly introduced concepts should be unified with existing concepts of the application ontology. Web 2.0 approaches like “tag clouds” showing the use of already defined concepts in a cloud or the use of auto-completion during the editing of the wiki text can help to re-use already existing concepts, but generally will not eliminate the problem. At the moment, we offer auto-completion in the edit pane of the wiki where typed names are auto-completed to the names of concepts known in the ontology. Furthermore, we are discussing the integration of a tag cloud to show the bold elements of the ontology at a prominent place in the wikiview.

For the analysis of the *existing* ontology we use visualization techniques: Cluster maps [8] have been introduced to visualize the structure of ontologies but can also be successfully applied on the ontology usage task described above. In [1] we adopted the cluster map approach to visualize the uniform use of findings for a given set of solutions in a “single knowledge base setting”, i.e., analyzing the use of concepts in a single knowledge base. In the context of a knowledge wiki the visualization of multiple knowledge bases is required, since every wiki page defines its own knowledge base. The cluster map then shows bubbles containing findings that are commonly used by a number of knowledge bases, and single nodes representing the knowledge bases of the corresponding wiki pages. If two wiki pages KB1 and KB2 use the same set of findings then both wiki pages

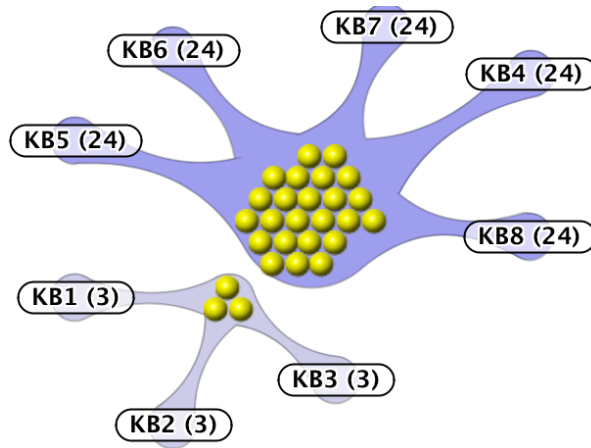


Figure 7: A ClusterMap showing a well-structured knowledge wiki, where the knowledge bases ( $KB_i$ ) have a high share of common findings.

nodes are connected with the joint bubble. In Figure 7 an example of a knowledge wiki visualization is shown. Here the wiki pages KB1, KB2, and KB3 share the same set of findings, whereas KB4 to KB8 share another set of common findings that is disjoint from the finding set of KB1 to KB3. In this example, we can see that the application ontology is used in a nearly uniform way, i.e., most wiki pages use the same concepts describing their embedded knowledge.

In contrast, Figure 8 shows the state of the BIOLOG wiki after an evolutionary growth phase. We can see that almost every wiki page has defined its own terminology and concepts are only shared by a few wiki pages. As shown in the upper left part of Figure 8, for example the pages *Outdoor recreation* and *Pest control bird insec...* share eight common concepts.

For finding similar concepts that *could* be shared we propose the following manual inspection technique: When clicking on a concept in a bubble, then all concepts in the entire graph that match the clicked concept in a definable way are highlighted. The simplest matching algorithm identifies concepts with a similar name, for example by computing the Levenshtein distance of the two names. More elaborated matching algorithms can also be applied and implemented, cf. [6].

Thus, (nearly) identical concepts in other knowledge bases can be found and unified in further steps. In summary, the proposed visualization is a starting point for the analysis of a knowledge wiki and denotes a helpful support for refactoring decisions.

### 3.2.2 Refactorings for Knowledge Unification

Based on the analysis of the wiki content a set of refactoring methods can support the developer when reorganizing the wiki. We discuss useful methods for unifying knowledge and concepts of the wiki in the following. In the concept of the paper we cannot discuss a complete set of meaningful refactorings but we focus on the two methods that we think to be most frequently applied in the current state of the project.

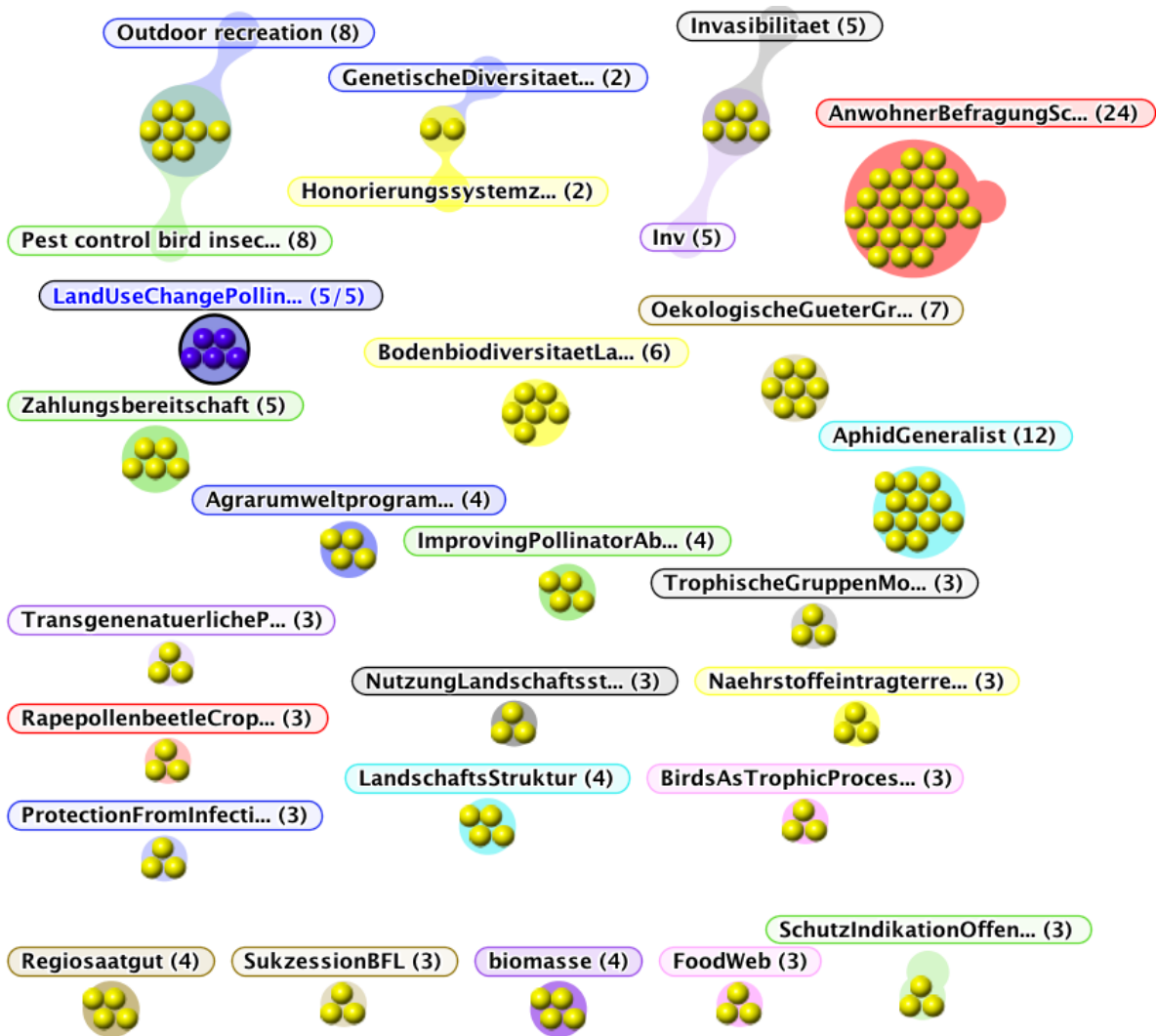


Figure 8: A ClusterMap visualization of an unorganized state of the BIOLOG Wiki.

**Rename Concept** Sometimes contributors create a new concept although an equivalent concept (named differently) already exists in the wiki. The equivalence can be easily expressed for example by the OWL construct `equivalentClass`, which aligns the two concepts. However, often it is more useful to unify the *names* of the concepts, too, i.e., by renaming one concept to the name of the other concept. Such a renaming refactoring can affect a large number of wiki articles and knowledge bases, respectively. Therefore, a preview of the intended changes in the wiki and their resulting renaming is very useful. In Figure 9 a renaming operation is shown for a concept *Training Goals* that provides a range of predefined answer choices. The rename operation considers the renaming of the answer choice *Self defense* to *self defense* and previews the expected changes in the article for each wiki page (separated by rulers).

Expectations	
<input type="checkbox"/>	loosing weight -- Stress alleviation -- <b>Self defense</b> -- Rehabilitation -- Staying fit -- Train
	loosing weight -- Stress alleviation -- <b>self defense</b> -- Rehabilitation -- Staying fit -- Train
Jogging - 2 hits	
<input type="checkbox"/>	Arms, Muscles = Back, Training Goals = <b>Self defense</b> , Training Goals = Rehabilitation, Training
	Arms, Muscles = Back, Training Goals = <b>self defense</b> , Training Goals = Rehabilitation, Training
<input type="checkbox"/>	Arms, Muscles = Back, Training Goals = <b>Self defense</b> , Training Goals = Rehabilitation, Training
	Arms, Muscles = Back, Training Goals = <b>self defense</b> , Training Goals = Rehabilitation, Training
Swimming - 2 hits	
<input type="checkbox"/>	Goals = loosing weight, Training Goals = <b>Self defense</b> , Training Goals = Train Coordination,
	Goals = loosing weight, Training Goals = <b>self defense</b> , Training Goals = Train Coordination,
<input type="checkbox"/>	= Stress alleviation, Training Goals = <b>Self defense</b> , Training Goals = Rehabilitation, Training
	= Stress alleviation, Training Goals = <b>self defense</b> , Training Goals = Rehabilitation, Training

**Warning:** Large scale changes can be done quickly, verify your selections carefully before replacing content!

Figure 9: A preview page of the refactoring of the concept *Training Goals*; here a value of the concept is renamed from *Self defense* to *self defense*.

**Coarsen/Extend Values Range** This refactoring changes the possible values of a choice question, i.e., a question with a pre-defined set of values. It was described in the context of the refactoring of single knowledge bases in [5]. The refactoring is especially useful before applying a *Rename Concept*: When aligning two almost equivalent concepts, their value ranges need to be unified to a common set of values before the actual concepts can be renamed.

Consider the following example: Article 1 contains a concept *temperature* with the possible values  $\{low, normal, high, very\ high\}$ . Article 2 defines a similar concept *temp* with the possible values  $\{low, normal, high\}$ . The developer decides to unify the concepts *temperature* and *temp*, but needs to unify the value range first. In our example, it would be possible to reduce the value range of *temperature* by the value *very high*. Then the concept *temp* can be easily renamed to *temperature*.

## 4 Conclusions

In this paper we described the semantic knowledge wiki KnowWE and introduced alternative markups for the effective elicitation of problem-solving knowledge. The markups are very simple, thus enabling even untrained users to contribute to a wiki application. We also showed that a distributed knowledge acquisition process will yield an unorganized knowledge structure (aka knowledge soup) and requires a thorough refactoring phase, for example when following the SER process model proposed by Fischer [7]. In order to simplify the complex task of the refactoring of a wiki we discussed useful methods like *Rename Concept* and *Coarsen/Extend Value Range*, that help to unify knowledge

concepts in the wiki. In our experience the unification of knowledge elements is the most frequent and complex operation in a knowledge wiki.

## References

- [1] J. Baumeister, M. Menge, and F. Puppe. Visualization Techniques for the Evaluation of Knowledge Systems. In *FLAIRS'08: Proceedings of the 21th International Florida Artificial Intelligence Research Society Conference*, pages 329–334. AAAI Press, 2008.
- [2] J. Baumeister and F. Puppe. Web-based Knowledge Engineering using Knowledge Wikis. In *Proceedings of Symbiotic Relationships between Semantic Web and Knowledge Engineering (AAAI 2008 Spring Symposium)*, 2008.
- [3] J. Baumeister, J. Reutelshoefer, K. Nadrowski, and A. Misok. Using Knowledge Wikis to Support Scientific Communities. In *Proceedings of 1st Workshop on Scientific Communities of Practice (SCOOP)*, Bremen, Germany, 2007.
- [4] J. Baumeister, J. Reutelshoefer, and F. Puppe. Markups for Knowledge Wikis. In *SAAKM'07: Proceedings of the Semantic Authoring, Annotation and Knowledge Markup Workshop*, pages 7–14, Whistler, Canada, 2007.
- [5] J. Baumeister, D. Seipel, and F. Puppe. Refactoring Methods for Knowledge Bases. In *EKAW'04: Engineering Knowledge in the Age of the Semantic Web: 14th International Conference, LNAI 3257*, pages 157–171, Berlin, Germany, 2004. Springer.
- [6] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, Berlin, 2007.
- [7] G. Fischer. Seeding, Evolutionary Growth and Reseeding: Constructing, Capturing and Evolving Knowledge in Domain-Oriented Design Environments. *Automated Software Engineering*, 5(4):447–464, 1998.
- [8] C. Fluit, M. Sabou, and F. van Harmelen. Ontology-based information visualization. In *Visualizing the Semantic Web*, pages 36–48. Springer, 2006.
- [9] M. Krötzsch, D. Vrandečić, and M. Völkel. Semantic MediaWiki. In *ISWC'06: Proceedings of the 5th International Semantic Web Conference, LNAI 4273*, pages 935–942, Berlin, 2006. Springer.
- [10] C. Lange. SWiM - A Semantic Wiki for Mathematical Knowledge Management. Technical Report 5, Jacobs University, Bremen, Germany, march 2007.
- [11] J. A. Reggia, D. S. Nau, and P. Y. Wang. Diagnostic Expert Systems Based on a Set Covering Model. *Journal of Man-Machine Studies*, 19(5):437–460, 1983.

- [12] J. Reutelshoefer, J. Baumeister, and F. Puppe. Ad-Hoc Knowledge Engineering with Semantic Knowledge Wikis. In *SemWiki'08: Proceedings of 3rd Semantic Wiki workshop - The Wiki Way of Semantics*, 2008.
- [13] S. Schaffert. IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In *STICA'06: 1st International Workshop on Semantic Technologies in Collaborative Applications*, Manchester, UK, 2006.
- [14] J. F. Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks/Cole Publishing Co., Pacific Grove, CA, USA, 2000.