Semantic Wiki Refactoring. A Strategy to Assist Semantic Wiki Evolution

Martin Rosenfeld, Alejandro Fernández and Alicia Díaz *

LIFIA, Facultad de Informática,
Universidad Nacional de La Plata, Argentina
{martin.rosenfeld,alejandro.fernandez,alicia.diaz}@lifia.info.unlp.edu.ar

Abstract. The content and structure of a wiki evolve as a result of the collaborative effort of the wiki users. In semantic wikis, this also results in the evolution of the ontology that is implicitly expressed through the semantic annotations. Without proper guidance, the semantic wiki can evolve in a chaotic manner resulting in quality problems in the underlying ontology, e.g. inconsistencies. As the wiki grows in size, the detection and solution of quality problems become more difficult. We propose an approach to detect quality problems in semantic wikis and assist users in the process of solving them. Our approach is inspired by the key principles of software refactoring, namely the cataloging and automated detection of quality problems (bad smells), and the application of quality improvement transformations (refactorings). In this paper we discuss the problem of evolving semantic wikis, present the core model of our approach, and introduce an extensible catalog of semantic wiki bad smells and an extensible toolkit of semantic wiki refactorings.

1 Introduction

Semantic Wikis [1] enhance the functionality of wikis with mechanisms to express semantic for the content in the wiki, in the form of *semantic annotations*. The synthesis of all semantic annotations in a semantic wiki defines its *ontology*.

Wikis evolve as a result of the collaborative effort of its users [2]. However, the uncoordinated edits of users may result in semantic wikis with poor quality regarding the underlying ontology. Thus, assistance needs to be provided to check that quality criteria are met and to minimize the effort of improvement.

In software engineering, refactoring techniques are applied to improve programs quality. The term refactoring [3] refers to the process of making persistent and incremental changes to a system's internal structure without changing its external behavior, yet improving the quality of its design. Refactoring is based on two key concepts: *bad smells*, that are an informal still useful characterization of patterns of bad source code, and *refactorings*, which are piecemeal transformations of source code that keep the semantics while removing (totally or partly) a bad smell.

* This work was partially funded by: the PAE 37279-PICT 02203 which is sponsored by the ANPCyT, Argentina.

A strategy for refactoring consists in the following components: a toolbox of refactorings, a catalog of bad smells, and detailed instructions on how to apply refactorings to remove each smell. Additionally, automated tools for the detection of bad smells and refactoring editors reduce the effort of refactoring and the chance of introducing errors. Environments in which refactoring is a mature practice (a culture), such as Smalltalk, usually offer these powerful tools.

Inspired by the principles of refactoring, this paper explores the use of such strategies to assist the evolution of semantic wikis and incrementally improve their quality. In this order, we give definitions for *Semantic Wiki Bad Smell* and *Semantic Wiki Refactoring*. Then, we adapt each of the components of a refactoring strategy to the context of semantic wikis. This includes a toolbox of semantic wiki bad smells, and a catalog of semantic wiki refactorings with their instructions on how to remove bad smells. Additionally, we discuss how tools can be used to automate the detection of bad smells and refactoring operations.

The structure of this paper is the following. We discuss in detail the problem of achieving quality in evolving semantic wikis in Section 2. Afterwards, in Section 3, we describe the state of the art in works to assist the evolution of semantic wikis. In Section 4, we present our approach that applies the ideas of software engineering refactoring in the context of semantic wikis. Finally, in section 5 we present the conclusions and future work.

2 The problem of evolving semantic wikis

The structure of a wiki, as well as its content, evolves as a result of the collaborative effort of the wiki users. Usually, wikis are created with very few or no structure in advance, so that users adjust it as a necessity to express knowledge, ideas, opinions, etc. Mader [4] suggests that the wiki creator should not try to guess the structure of a wiki in advance, but let it evolve into the optimal organization of information as people use it.

When the wiki is semantic, the wiki evolution also affects the formal representation of the wiki content, i.e. the underlying ontology. Thus, the ontology emerges with the wiki, as users add semantic annotations to the wiki content. Kousetti et. al. [5] use the term *Ontology Convergence* to mean the process of the implicit ontology evolving to a single model.

The problem of evolving wikis consists in assuring a good wiki quality through all the stages of the wiki evolution. If the evolution is not controlled, it is very possible that the wiki evolves in a chaotic manner. Many problems can arise as the result of multiple users collaboratively and incrementally editing a wiki. In traditional wikis, these problems are usually related to the following quality metrics: readability, structure, navegability, completeness, consistency.

Semantic wikis aim to improve the quality of traditional wikis by allowing to add semantic knowledge to the wiki content. However, they cannot assure that the wiki evolves in a correct manner and that a good ontology convergence is achieved. Kousetti et. al. [5] describe the following quality metrics for the semantic wiki's implicit ontology:

- Consistency: no sentence can be contradicted.
- Completeness: anything that needs to be in the ontology is explicitly defined or it can be inferred from other defined definitions and axioms.
- Conciseness: does not contain unnecessary definitions or redundancies.
- Expandability: you can easily add more knowledge without requiring to make major changes to the existing structure.
- Sensitiveness: the ontology is more sensitive if small changes can alter easily how well-defined a definition is.

Let's take a motivating example of a semantic wiki being employed to describe geographical places. In an early stage of evolution, a user creates a category "City". Later, another user creates a category "Capital city", attempting to form a more specific group for cities that are also capitals. However, this user does not realize that this category defines a subset of the resources of category "City" and, consequently, should be a subcategory of it. The lack of this semantic relation between the two categories results in many problems. First of all, it affects the conciseness of the semantic wiki. Every time a user creates a page for a city that is also a capital city, he will have to categorize it with both categories, so that no semantic knowledge is lost. Secondly, it affects the semantic wiki completeness. A user asking the wiki for a list of cities will not get as result the resources categorized with category "Capital City" but not with category "City".

The example above exposes that problems can arise while the evolution of semantic wikis takes place. Such problems are finally perceived as a lack of quality in the product, thus jeopardizing the success of the wiki. This results in the necessity of an extra work, that consists in periodically checking for problems in the wiki quality, and making the corresponding modifications. This work is commonly called "wiki gardening"[1]. However, as the wiki becomes longer, with many pages and semantic annotations, it becomes more difficult to detect quality problems and solve them appropiately.

In this context, where shared ownership and evolution of structure and content are a plus, assistance needs to be provided to check that quality criteria are met and to minimize the effort of improvement.

3 State of the art

Wikis are groupware. Peter and Trudy Johnson-Lenz [6] identify two prevailing approaches to groupware: *mechanism*, the use of explicit forms and procedures, and *context* or *open space*, to allow groups to self organize. Wikis are by nature closer to the later approach. There are semantic wiki tools, such as *Project Halo*[2], that support the definition of the ontology upfront (thus stressing the "mechanism"' approach). Those wikis are usually called *Semantic Data Wikis*. In contrast, our research focuses on wikis created with very few or no structure in advance, so that users incrementally create the ontology.

[1] http://www.wikisym.org/ws2008/index.php/What_are_the_tasks_of_a_wiki_gardener%3F
[2] http://www.mediawiki.org/wiki/Extension:Halo_Extension

In the field of ontologies, the problem of ontology evolution is one of the current topics of the research agenda. Djedidi et. al. [7] and Haase et. al. [8] focus on consistency achievement when a change in the ontology is performed. They employ resolution mechanisms to recover from four levels of inconsistencies: structural, logical, conceptual and domain dependent. Moreover, the former presents a list of metrics to evaluate the quality of an ontology. Baumeister and Seipel [9] describe a set of anomalies in the design of the ontology that may affect its mantainability, usability and understandability, and propose refactoring methods to eliminate them. They distinguish four categories of anomalies: redundancy, circularity, inconsistency and deficiency. Although these approaches make significant contributions to the field of ontologies, the domain of semantic wikis presents new challenges. Firstly, the collaborative way of editing a wiki increments the possibilities of generation of different kind of anomalies. Secondly, the combination of semantic annotations with wiki's tacit knowledge (i.e. text, images, etc.) has to be considered.

An approach to assist the evolution of semantic wikis is MOCA [5], which is a Semantic Mediawiki (SMW) [10] extension. It is a support system that assists wiki authoring and contribution to the background ontology. MOCA provides assistance in the edit page of the wiki, giving help with recommendations for types using the background ontology, and insertion of annotations without knowledge of the syntax. A similar approach is taken by the *Project Halo*, which is also an extension of SMW that provides intuitive graphical interfaces that facilitate the authoring, retrieval, navigation and organization of semantic data.

Although these two works assist the evolution of semantic wikis by helping users in the creation of content, they do not aim to neither find quality problems nor solve them. SMW+[3] is actually an approach to accomplish part of this. SMW+ is a semantic wiki built on top of SMW, aimed at the enterprise market. In addition to assisting users in authoring using the Halo Extension, SMW+ offers gardening functionality. It comprises a set of programmable tasks (called bots) to automate or assist users in the process of improving the quality of the wiki content. For example, there are bots to find pages without annotations, or to find artifacts on schema or annotation level which indicate a bad modeling.

Although the contribution of the gardening functionality of SMW+ should be recognized, it is strongly biased towards the identification of quality problems. It does not provide support for the implementation of the changes required to remove them and, thus, to improve the quality of the semantic wiki.

4 Refactoring semantic wikis

In this section we will see how each of the components of a refactoring strategy, that were described in section 1, are applied to the context of semantic wikis. But first of all, in section 4.1, we will need to define the model of the artifact that will serve as the base to define each of them.

[3] http://wiki.ontoprise.de/smwforum/index.php/MainPage

4.1 Core model

In order to define refactorings and bad smells, we need a model of a semantic wiki that will serve as a base. We call it the *core model*, which is shown in figure 1. It is defined as an OWL ontology, and we use UML diagrams to visualize it.

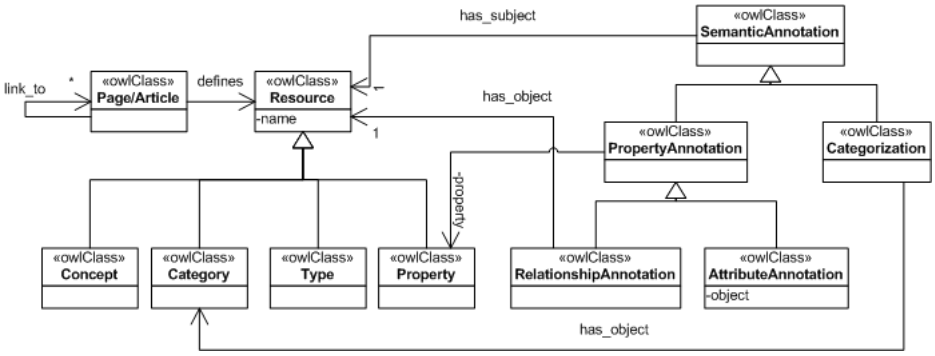

Fig. 1: Core Model: a conceptual model of a semantic wiki

The core model is a conceptual model of a semantic wiki and is inspired on the knowledge base of SMW. However, it can be customized and extended to support new features, so that new refactorings and smells could be defined in terms of them.

A *Resource* is anything that can be described in a wiki: *Concept*, that represents individuals in the wiki; *Category*, which allows to group resources; *Property*, that describes the semantic of relationships between resources; and *Type*, that is used to specify the object type of properties. Each of the resources can have a wiki *Page* or *Article* that describes it. Pages can be also directly related to other pages using untyped links.

Semantic Annotation is subclassified in Categorization and Property Annotation. A *Categorization* describes a relation of membership of a resource in a category. A *subcategory relationship* between categories is defined when the subject resource of a categorization is a category. In this case, the subject category is a *subcategory* of the object category.

A *Property Annotation* is composed by three elements: a *subject*, that can be any resource; a *property*, which is an individual of Property; and an *object*. *Relationship Annotations* are those property annotations whose object is another resource, while the object of *Attribute Annotations* are end values of a data type. Categorizations could be restated as relationship annotations in which the property describes the *belongs to* semantic, and the object is allways a category.

We assume the existence of a set of built-in properties with special meanings. One of them is the *Subproperty of* Property. When used as the property of a relationship annotation between two properties, it describes a subproperty rela-

tionship between the subject property and the object property. Another built-in property is the *Has type* Property that indicates the object Type of a Property.

4.2 A toolbox of semantic wiki refactorings

Semantic wiki refactorings are transformations in the semantic structure of a semantic wiki knowledge base, that allow to improve the implicit ontology quality. Refactorings describe an ordered way of appropiately performing such transformations, so that new inconsistencies, redundancies and other quality problems are avoided. Each refactoring is defined by: a *name*; a *description*, which summarizes the refactoring in informal language; and the *mechanics*, which are a series of mechanical transformations in the model of the wiki.

In the following we describe in detail the refactoring called *Create subcategory relationship* as an example. As it is not the objective of this paper to define a complete catalog of refactorings, we will later list other refactorings for which we will only mention their name and description.

Create subcategory relationship refactoring describes a transformation in which an existent category turns to be a subcategory of another one. Defining such relationship helps improve the quality of search results, the conciseness and completeness of the semantic wiki.

Following the example described in section 2, applying this refactoring would mean to make "Capital City" a subcategory of "City". The first step to appropiately accomplish this, must consist in actually making "Capital City" subcategory of "City". This is the basic step to create the subcategory relationship. Secondly, the categorizations of category "City" in each resource that also belongs to category "Capital City" have to be removed. This is because it is now implicit that every resource that belongs to "Capital City" also belongs to "City". Applying this step avoids redundant categorizations. The last step consists in eliminating redundancies in the categorization chain. Supose the existence of a category "Geographical place", that is supercategory of both categories. Then, the subcategory relationship that states that "Capital City" is subcategory of "Geographical place" is now redundant and, thus, has to be eliminated.

The following is the formalization of this refactoring. Applied to the example, the category "City" plays the role of *supercategory*, "Capital City" is the *subcategory*, and "Geographical place" is the *super-supercategory*:

Name Create subcategory relationship.
Description Make an existent category *subcategory* subcategory of another existent category *supercategory*.
Mechanics

1. Make *supercategory* supercategory of *subcategory*.
2. For each resource that belongs to *subcategory* remove its categorization of *supercategory*.
3. For each category *super-supercategory* that is supercategory of *subcategory*: if it is also a supercategory of *supercategory*, remove it as supercategory of *subcategory*.

Note that the steps defined in the mechanics of the refactoring are not bound to any semantic wiki implementation. The concrete actions that has to be done to actually perform the steps, depend on the way each semantic wiki implementation implements each of the semantic wiki features. In this way, refactorings are able to be reused in any semantic wiki implementation.

Table 1 presents a list of other possible semantic wiki refactorings. This list can be extended and customized. Furthermore, if the implementation of the semantic wiki supported new features, it could be extended to make use of them.

Name	**Description**
Move annotation	Change the subject of an annotation from one resource to another.
Unify categories	Unify two existent categories categoryA and categoryB. A new category categoryC is created and replace both of them. Existent categorizations of the two unified categories have to be replaced to the new one.
Split property	Split a property propertyA into two new properties propertyB and propertyC. The semantic annotations that have propertyA as property, have to appropiately change it to propertyB or propertyC.
Extract concept	A concept conceptA describes more than one real concept. A new concept conceptB is created. All the semantic annotations that have conceptA as subject or object, have to be placed appropiately.
Create subproperty relationship	Create a subproperty relationship between two existent properties.
Rename concept	Rename a concept and, consequently, the page that describes it. All the semantic annotations and untyped links that point to the concept, have to be updated.
Remove category	Remove an existent category. All its categorizations have to be also removed.

Table 1: List of semantic wiki refactorings

4.3 A catalog of semantic wiki bad smells

Refactorings describe *how* to appropiatelly perform transformations in the wiki. However, they say nothing about *when* they should be applied. Fowler [3] afirms that "deciding when to start refactoring, and when to stop, is just as important to refactoring as knowing how to operate the mechanics of a refactoring".

Defining bad smells may help determine when to apply each refactoring. A *semantic wiki bad smell* is a symptom in the semantic structure of a semantic wiki that possibly indicates a deeper problem, and suggests that a refactoring should be applied. It must be said that the detection of a bad smell does not necessarily implies a real problem. It must be analyzed in the current context and decide whether it should be applied or not a refactoring.

Each bad smell is defined with: a *name*; a *description*, which describes the bad smell in informal language; the *related refactorings* that should be applied

to remove the smell; and a *detection mechanism*. The detection mechanism may simply involve the execution of a query to the wiki knowledge base, or may apply more sophisticated methods such as data minning techniques.

As with refactorings, it is not the objective of this paper to define a complete catalog of bad smells. We will first describe as an example the *Twin categories* bad smell and then present a preliminar list of other possible ones.

The *Twin categories* bad smell describes the case when two categories appear together in categorizations very frequently. The first reason why this bad smell can be detected, is if two categories have the same semantic but different names. This case is intrinsic to the collaborative way of semantic wikis. This situation can arise because a user does not know a category already exists, or because of users belonging to different professional stuff or speaking different languages. The consequence of this problem is that a category is duplicated.

The second possible cause is that the twin categories define a subcategory relationship. The example presented in section 2 describes this situation. Because of the lack of the subcategory relationship between the categories "Capital City" and "City", every time a resource is categorized with category "Capital City", it should be also categorized with category "City". To remove the bad smell in this case, the "Create subcategory relationship" refactoring should be applied.

The following is the full definition of this bad smell:

Name Twin categories

Description Categorizations of two categories categoryA and categoryB appear together very frequently.

Related Refactorings

1. Unify categories
   Cause: The two categories describe the same category.
   Example: categoryA = "LIFIA", categoryB = "LIFIA Lab"
2. Create subcategory relationship
   Cause: The two categories describe a subcategory relationship.
   Example: categoryA = "Capital city", categoryB = "City"

Detection Mechanism In this case, we decided to use a semantic query as the detection mechanism. The semantic query is expressed in SPARQL query language, and is based on "SPARQL 1.1 Query"[4] specification.

```
SELECT DISTINCT ?categoryA ?categoryB
WHERE {  ?categorizationA has_subject ?subjectA
         ?categorizationB has_subject ?subjectA
         ?categorizationA has_object ?categoryA
         ?categorizationB has_object ?categoryB.
FILTER ((?categorizationA != ?categorizationB)
         && (?categoryA != ?categoryB))
      }
GROUP BY ?categoryA ?categoryB
HAVING (count(*) > PARAM)
```

Table 2 presents a list of other possible semantic wiki bad smells.

[4] http://www.w3.org/TR/2009/WD-sparql11-query-20091022/

Name	Description
Concept too categorized	A concept belongs to too many categories. This symptom may expose the fact that the concept describes more than one real concept.
Divergent property	Instances of a property diverge in range and domain. A property is used with many semantics.
Twin properties	Annotations of two properties appear together in the same resources very frequently.
Resource with no semantic annotations	A resource is not subject of any semantic annotation.
Large category	A category is object of too many categorizations.

Table 2: List of semantic wiki bad smells

4.4 Automating the detection of bad smells and refactoring operations

Automating the detection of smells is a necessity as the wiki grows in size and semantic knowledge. Defining a detection mechanism such as a semantic query for each smell, makes it possible to find all the occurrences of a bad smell automatically. Integrating a tool for the detection of bad smells in the semantic wiki would allow users to look for quality problems with less effort.

The mechanics of the refactorings describe an ordered way of applying transformations, consisting in a series of simple actions. However, it could become tedious and time-consuming to execute those actions by hand. Moreover, the chances of making mistakes and introducing new errors are increased. For wiki refactoring to be a productive strategy to assist semantic wiki evolution, effort and risk must be minimized.

Fortunately, many times it is possible to have a refactoring automation tool. Take for example the *Create subcategory relationship* refactoring. You have several changes to make and check for correctness if you do it by hand. With a refactoring automation tool, one simply selects both categories (subcategory and supercategory) and launches the refactoring. The refactoring tool applies all the steps of the mechanics as part of the same single transaction. The whole process takes seconds instead of several minutes. Moreover, the refactoring operation ensures that no action is lost and no bugs are introduced.

5 Conclusions and future work

Semantic wiki refactoring is an approach to detect quality problems in semantic wikis and assist users in the process of solving them. It is inspired by the key principles of software refactoring. We have discussed the problem of evolving semantic wikis, presented the core model of our approach, and introduced extensible catalogs of semantic wiki bad smells and refactorings.

As future work to complete and extend this investigation, it remains to complete the catalogs of semantic wiki bad smells and refactorings. A categorization

should be defined to achieve a better understanding. In other respects, the tools to automate the semantic wiki refactoring operations and semantic wiki detection of smells should be developed. It will allow to carry out experimental evaluations to check the effect of a refactoring strategy in the evolution of semantic wikis.

In other sense, as wikis have a strong social component, it can be investigated the social factor in a semantic wiki refactoring strategy. In this order, collaborative detection of smells and collaborative testing of refactorings should be studied. The first one arises as a necessity because some bad smells are difficult to be detected by automated detection mechanisms, e.g. a bad smell to detect inappropiate names for categories. We propose a social detection mechanism in which users could collaboratively agree the presence of a bad smell. The second takes the idea from software engineering agile methods. They advocate the use of automated unit testing to ensure that no bugs were introduced and the success of a refactoring. Adapting this idea, we propose a social testing in which wiki users could collaboratively state if a refactoring was successful or not.

Finally, future work will deal with refactoring of wiki's tacit knowledge. Tacit knowledge should be considered in the refactoring operations, and could also be the source of bad smells.

References

1. Schaffert, S., Bry, F., Baumeister, J., Kiesel, M.: Semantic wikis. IEEE Software **25**(4) (2008) 8–11
2. Leuf, B., Cunningham, W.: The Wiki way: quick collaboration on the Web. Addison-Wesley (2001)
3. Fowler, M.: Refactoring: Improving the Design of Existing Code. Addison-Wesley, Boston, MA, USA (1999)
4. Mader, S.: Wikipatterns. Wiley Publishing (2007)
5. Kousetti, C., Millard, D., Howard, Y.: A study of ontology convergence in a semantic wiki. In: WikiSym 2008. (2008)
6. Johnson-Lenz, P., Johnson-Lenz, T.: Post-mechanistic groupware primitives: rhythms, boundaries and containers. Int. J. Man-Mach. Stud. **34**(3) (1991) 395–417
7. Djedidi, R., Aufaure, M.A.: Onto-evoal an ontology evolution approach guided by pattern modeling and quality evaluation. In Link, S., Prade, H., eds.: FoIKS. Volume 5956 of Lecture Notes in Computer Science., Springer (2009) 286–305
8. Haase, P., Stojanovic, L.: Consistent evolution of owl ontologies. In: Proceedings of the Second European Semantic Web Conference, Heraklion, Greece. (2005)
9. Baumeister, J., Seipel, D.: Verification and refactoring of ontologies with rules. In: EKAW'06: Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management. (2006) 82–95
10. Krötzsch, M., Vrandecic, D., Völkel, M., Haller, H., Studer, R.: Semantic wikipedia. Journal of Web Semantic **5**(4) (2007) 251–261