Automatic Deployment of Semantic Wikis: a Prototype

Angelo Di Iorio[1], Marco Fabbri[1], Valentina Presutti[1], Fabio Vitali[1]

[1] Department of Computer Science, University of Bologna, Italy
{diiorio, mfabbri, presutti,vitali}@cs.unibo.it

Abstract. Semantic wikis simplify the creation, searching and management of content in a specific domain of interest. Although very powerful solutions exist for adding semantics to wikis, the authoring process of domain-oriented content still remains a manual and quite consuming task. We propose a different approach to deploy semantic wikis: the automatic delivery of a customized wiki for a given domain, taking as input an ontological description of that domain. WikiFactory is an application that takes these ideas to implementation, based on a strong distinction between the ontology designer, the content author and the graphic designer. Moreover WikiFactory is designed to be independent for a specific wiki clone and commit an abstract description of pages onto a wide set of wiki platforms. In this paper we present the early implementation of WikiFactory that automatically generates pages for MediaWiki.

1. Introduction

Wikis[2] are increasingly gaining importance among the web authoring tools, either as personal web sites, or as well-featured information systems supporting schools, universities and firms[16]. Among wiki clones, a leading role is played by semantic wikis, which combine the success of the wiki model, with the power of Semantic Web technologies. A Semantic Wiki[17] is a wiki enhanced in order to encode more knowledge than just structured text and hyperlinks, and to make that knowledge readable by machines too. They make it easy to manage, search and retrieve information among the wiki pages and entities. Several examples of semantic wikis can be cited: RDFWiki[12] provides users with a simple text-based interface to edit content and metadata and stores all data as RDF statements; SemanticMediaWiki[10] is an extension of MediaWiki (the wiki platform used by the WikiPedia community[18]) that allows users to add metadata understandable by automatic processes too; Rhyzome[14] allows users to express RDF statements through a simplified syntax called ZML, and many other projects were proposed by researchers in order to merge wikis with semantic web technologies.

A different point of contact between these research efforts can be also figured out: using semantic information in order to generate wikis, apart from annotating them. In particular, we propose to generate content for a wiki, taking in input an ontological description of the domain where that wiki will be used. Each domain, in fact, suggests a natural structure of a related wiki, describing clusters of pages, navigation paths but also templates for each page, or dynamic behaviors and so on. For instance, a wiki for

a university is supposed to have pages for courses, classes, professors, rooms, events, exams, and so on; each page is expected to express some information organized according to a given template. Moreover, a lot of repeated pages, repeated structures and repeated templates can be found. What usually happens is that a university employee writes the content of those pages one by one, filling them with the proper data, through an error-prone and time-consuming process.

In [5] we describe WikiFactory, a framework designed for the automatic generation of wikis from ontological descriptions. WikiFactory centres on an OWL description of a domain, written by different users with different skills and processed by an engine that translates such description into actual wiki pages. A first advantage of such approach is clear: it makes automatic, fast and easy the manual process described so far. But another aspect is equally important: WikiFactory does not produce only *wikis* but even a sort of *semantic wikis*. All pages are natively decorated with metadata, directly derivable from the input ontology: relations among entities in the domain can be easily mapped in relations among the wiki pages, as well as objects' properties can be transformed in metadata about those pages. The current implementation of the system does not store metadata yet, but it will be simple to include them into the final wiki, moving off the pool of semantic data behind the generation process.

More details about WikiFactory can be found in [5], where we discussed the rationale behind the system, the overall architecture and the goal of our model. In this paper we present a very early prototype of WikiFactory and discuss how the abstract model has been instantiated and implemented in a running application. The rest of the paper is structured as follows: section 2 gives a brief overview of the WikiFactory publishing model; section 3 illustrates how the current prototype works through a simple case study; section 4 discusses the internal architecture of the system and the conclusions depict possible evolutions of the overall project.

2. WikiFactory: a prototype for semantic wikis generation

WikiFactory prototype is a Java application aiming at demonstrating the feasibility and the potential of the WikiFactory's model. Although WikiFactory is designed to generate content for different wiki clones, the current prototype works on MediaWiki[11] and generates pages for that specific clone only. It is a very first implementation of a more complex architecture, which cover many issues about domain-oriented and reliable wikis.

The publishing model behind WikiFactory changes the classic wiki publishing approach since the creation of pages becomes an automatic effect of describing a domain, rather than a direct authoring process. The lifecycle of a common wiki is quite simple and straightforward: an administrator sets up a wiki software and its dependencies (as a web server or an external database), and later many users add content manually by creating and editing topics. Semantic wikis add a new dimension to this workflow, since users can also add metadata to the pages, during the editing phase.

The simplicity and speeding of such approach gave a great contribution to the intensive and widespread diffusion of wikis among Internet communities, companies

and organizations. On the other hand, it is still limited in terms of automation, since most of the authoring work still remains completely manual. Fig. 1 shows such a simple scenario:

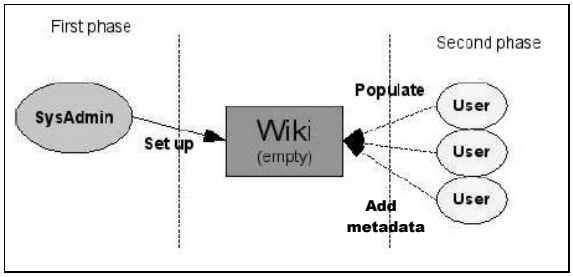

Fig. 1. The wiki publishing model

WikiFactory adds an intermediate phase to this process in order to automate the production of repeated pages and structures, by exploiting ontologies. Such improved version of the wiki publishing model is depicted in fig. 2.

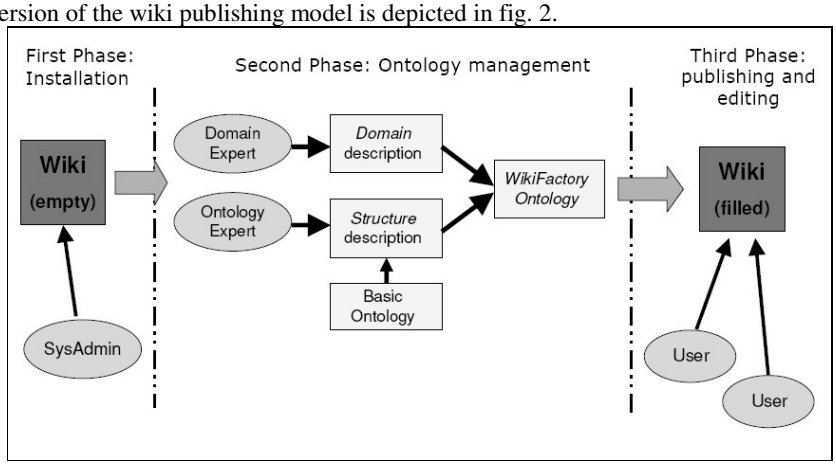

Fig. 2. The WikiFactory publishing model

The installation and editing steps do not change (although the automatic installation of the wiki environment is a requirement – with no high priority - of WikiFactory), while a new intermediary step shows users writing (or importing) an ontology and WikiFactory populating the final wiki with content extracted from this ontology. The core of the application is just the ontological description we name *WikiFactory (based) Ontology*, that describes everything needed by the WikiFactory engine in order to populate the final wiki.

Three main sub-components can be identified within such ontology:

- *WikiFactory Basic*: a core ontology supplying rules, constructs and objects used by the designers to define the rest of the ontology. Note that it is different from the whole assembled ontology, whose name is quite similar.
- *Domain Description*: a representation of the domain describing the entities and relations of the domain, and the data to be filled in the wiki.
- *Structure Description*: the actual description of the domain-oriented wiki, a model of topics' structures and templates.

The assembled ontology indicates how to populate the final wiki, by inserting the data provided by a domain-expert, and modelled by an ontology-expert. Two more actors, in fact, exist in that scenario:

- a *domain expert*, we name Bianca, i.e., an inexperienced user who adds content and uses the final wiki every day for carrying out her tasks;
- an *ontology expert*, we name Andrea, who adapts the requirements of the domain experts and actually produce the final ontology.

In order to explain the role played by these two experts and to discuss the structure of the ontology with more details, as well as the internal functioning of the system, we present a case study in the following section.

3. WikiFactory workflow: a simple case study

Consider a wiki used by a Computer Science Department of a University (CSD), say the University of Bologna. Such a wiki (CSD wiki) is supposed to have pages about professors, courses, classrooms, staff and so on. WikiFactory prototype is an early application, not yet mature to produce the whole CSD wiki, but it already provides the most relevant constructs in a flexible and extensible framework.

In the rest of the section we describe how the WikiFactory prototype can be used to produce a CSD wiki simply composed by a department home page linked to the list of the affiliated professors and, for each of them, a home page with some information and a list of courses he/she teaches. This example allows us to discuss two relevant goal of the application: (i) describing and committing a set of pages linked each other and (ii) describing and producing a single wiki page. As discussed in [5], the whole design is completed by an automatic deployment of scripts that support dynamic behavior, but that feature is not still supported in the current prototype.

3.1. Setting up a wiki target platform

According to the above mentioned schema, the first step consists of a common installation of a wiki platform: a system administrator from the University's technical staff receives a request for a wiki, say MediaWiki, to be installed on a department machine. He installs all required components and a new installation of MediaWiki is now available, with no content yet (apart from content already provided in the installer). It has to be noticed that among the requirements of the WikiFactory platform

there is also the automatic configuration and installation of the desire environment, i.e. the wiki clone. Nevertheless, such requirement now has less priority than others.

3.2. Describing a domain-oriented wiki

The second step of the process consists of producing the *WikiFactory (based) Ontology* collecting all data required to populate the final wiki. As discussed before different actors are involved in creating different parts of such ontology (actually the basic ontology is a built-in feature of WikiFactory).

3.2.1. What a domain expert does

Bianca is an employee of the Computer Science Department in charge of supplying information about professors and courses of the department. Such information has to be encoded in RDF statements saying that "a course named 'Web Technologies' exists and it's taught by Fabio Vitali" or "Fabio Vitali is a professor", or "You can contact Fabio Vitali by email at fabio@cs.unibo.it or by phone at 0512094872".

Obviously we cannot expect that Bianca fills manually such RDF document, but we need an automatic process that produces such document. Currently WikiFactory provides her a very simple interface shown in fig. 3. The study of the Graphical User Interface (GUI) is a sensitive aspect, an important requirement of WikiFactory. Nevertheless, its definition has not been approached deeply and comprehensively yet because it depends on the definition of all functionalities WikiFactory is intended to support.

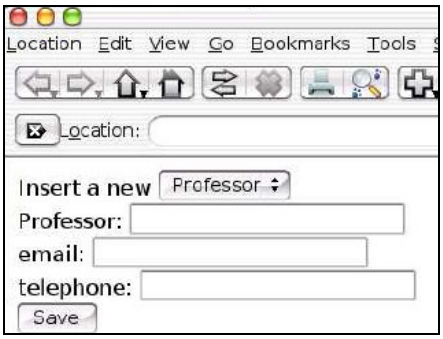

Fig. 3. The WikiFactory Interface for Bianca

The relevant aspect is that Bianca perceives such task as a raw insertion of data, but she is actually populating the ontology (i.e, completing the *domain description*). A clarification is needed at this point: even such description could be further divided in two sub-components, the *generic description* of a domain, and *specific description* of an instance of that domain. Consider the CSD case study: a generic description says that a computer science department has a set of professors and each of them can be described by a set of personal information, including a list of courses he teaches; a

specific description says that the computer science of the University of Bologna has a specific list of professors, including Fabio Vitali, that teaches "Web Technologies" and can be reached by a specific email address or phone number.

One of main important activities we are planning for WikiFactory is just studying the automatic production of the above mentioned interface, from a generic domain description: the interface shown in fig. 3 has been created manually, but it could have been automatically created from a pre-existing ontology about the university domain.

3.2.2. What an ontology expert does

Andrea is an ontology expert in charge of producing the *Structure Description* of the CSD wiki. He works with specific tools such as Protégé[13] and actually writes RDF statements listing the pages of the wiki, describing their connections with the domain entities and their internal structure. Another requirement of WikiFactory is interoperability with Protégé.

The main element provided by the WikiFactory Basic Ontology is called *TreeOfTopic*. A TreeOfTopic represents a set of wiki pages (or a single one) and, as expected, will be instantiated into corresponding page(s) in the final wiki. In turn, a TreeOfTopic is composed by one or more optional *Iterator(s)* and a *TopicTemplate*.

The element Iterator is a very general purpose structure indicating a class of individuals within the domain description: it can be used to tell WikiFactory to generate a page for every instance of that class, and to link that page to the current one. Andrea creates an instance of TreeOfTopic named Professors, containing a Professor Iterator; this Iterator has a specific property pointing to the instances of professors within the domain description. At the end of the process, for each professor included in the list filled by Bianca, a new link will be added to the Professors page.

The element *TopicTemplate* is used to declare the fragments composing a page. The basic assumption is that a small set of components can be identified, able to capture the internal structure of any wiki page, regardless of the wiki platform or the subject of the page. In [4] few patterns are identified able to express the content of any web page too: (classified) paragraphs, headings, tables, records and few other things. What Andrea does is simply describing which components are included in a given page and which text each component is made of. At the end of the process, these elements will be re-flowed and formatted according to the layout of the final wiki. Actually, the current prototype of WikiFactory allows Andrea to specify only the title and the whole body of a page, but we are working on more sophisticated templating languages and solutions.

A very simple rule is used within the WikiFactory prototype: each instance of a word preceded by the character '$' will be automatically replaced by the value of the matching property defined in the Domain Description. Then, the following text fragment "This is *$resource_title* personal home page. You can contact him/her by phone calling *$telephone_number* or by sending an email to *$email*" will be filled with the right data, calculated by the Iterator, and previously inserted by Bianca. Obviously, such a simple templating language cannot be enough for all the real-world scenarios (consider for instance multiple phone numbers or emails), so that we are investigating more complex solutions: particularly interesting are some Java templating engines that could be easily integrated in WikiFactory, such as FreeMarker[6] or Velocity [1].

3.3. Instantiating a wiki

The final step of the process shows the WikiFactory engine translating the ontology created by Bianca and Andrea into an actual wiki. According to the semantics of the Iterators and Templating operators, the engine collects all data inserted by Bianca and put them into wiki pages. Fig. 4 shows how the information about a professor are presented in MediaWiki.

Fig. 4. A simple page on MediaWiki created by WikiFactory

4. A modular java application

The WikiFactory prototype is a java application composed by different modules that work together in order to deploy content on MediaWiki, taking in input the Wiki-Factory Ontology described so far. Fig 5 summarizes the architecture of the system:

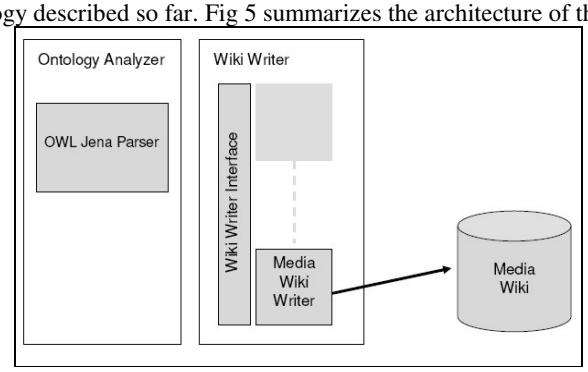

Fig. 5. The architecture of WikiFactory Application

Two main components can be identified:

- the *Ontology Analyzer*: that parses the input ontology and collect data about what and how is being published.

- the *Wiki Writer*: that commits the changes reported by the Ontology Analyzer on the target MediaWiki installation.

The Ontology Analyzer includes external libraries to handle OWL documents, in particular the Jena Parser[8], a Java library developed by HP for the access and management of Semantic Web documents. While retrieving information, the analyzer notifies them to the WikiWriter in order to actually produce pages. As expected, the Wiki Writer exports method for: (i) creating wiki-pages according to a given structure (that is, writing the body text indicated in the ontology) and (ii) iterating such creation for all the elements pointed by the Iterator construct. At the end of the process, a commit operation stores information on the wiki clone.

Actually the Wiki Writer is an abstract component, that is a Java interface implemented by every specific wiki clone Writer. The WikiFactory MediaWiki Writer exploits a MediaWiki extension that allows a batch creation of pages, by submitting a text file to a PHP script by Jonathan Cutrer[3]. According to the output of the Analyzer such a file (which even collects many pages together) is written and sent to the MediaWiki target installation through an HTTP connection.

Different solutions can be implemented by different writers, such as writing directly on files, or posting data as a common page editing, or writing a DB or anything else: users can deploy the same wiki from the same ontology on different platforms, since the complexity is hidden within the system. Particularly interesting in this context are the standard Wiki XML-RPC [9] APIs, a set of interfaces based on XML-RPC technologies that allow any client to interact with a wiki, regardless of its internal implementation. We plan to further investigate these libraries in order to standardize the communication and to make WikiFactory completely independent from the target wiki platform.

5. Conclusions

The WikiFactory prototype has shown how a simple wiki can be deployed, taking in input an ontological description of its domain of interest. The system relies on a strong distinction between the roles of the users: a domain-expert has to simply insert data, without dealing with their actual formatting into a wiki page, while an ontology-expert has to describe structures, without dealing with the data filling. Most of the routine work is performed by the engine behind the scenes, so that the whole process is simple and automatic.

The prototype presented in this paper is still very much in its initial phases, but we plan many activities towards a complete maturation. First of all, we are working to improve the WikiFactory Ontology and, subsequently, the OntologyAnalyzer: apart from issues about parameterization, configuration and performances, we are working on a more complex set of objects (including tables, records and other fragments) useful to create complex pages, to support more wiki features and to handle dynamic behaviour. As expected, we also plan to code new Wiki Writers, producing content for many other wiki platforms e.g., TWiki[15] or many semantic wiki.

Another important issue raised while working on the prototype is supporting modular deployment. We think that it is a prior and important requirement of WikiFactory to allow users to deploy wiki modules. In fact, it is desirable to add new elements (e.g., structures, templates, topics) to a domain-oriented wiki already deployed without affecting the existing content.

Issues about the interface for non-expert users are being investigated too: ontologies could be also exploited to dynamically create user interfaces (such as web forms, stand-alone applications or anything else), flexible and not hard-coded within the system.

Finally, we plan to focus on a tricky issue never discussed so far: what does it happen on a wiki deployed by WikiFactory after its installation? The current prototype does not face such problem yet, since the editing phase is completely disconnected from the wiki deployment but a lot of interesting issues can be raised by studying the consistency between the ontological view of a wiki and its actual pages, as well as the techniques used to update both of these views. What we want to do is either propagating changes from the ontology into the wiki or, the opposite, updating the ontology according to the modifications on wiki content. This is a very difficult task and we haven't yet found a good solution but we consider it the most important development track of our research.

However, WikiFactory is a lively project, whose preliminary implementation has been described in this paper. More detailed and up-to-date information can be found in its wiki site, at the address http://swe.web.cs.unibo.it/WikiFactory/.

6. References

1. Apache Jakarta Project, "The Velocity Template Engine", http://jakarta.apache.org/velocity/.
2. Cunningham, W. & Leuf B.*The Wiki way*. New York: Addison-Wesley, 2001.
3. Cutrer J. "Mediawiki bulkpage Page Creator", http://meta.wikimedia.org/wiki/MediaWiki_Bulk_Page_Creator.
4. Di Iorio A., Gubellini D., Vitali F. "Design patterns for document substructures". In the *Proceedings of Extreme Markup Conference 2005*, August 1-5, 2005, Montreal, Canada.
5. Di Iorio A., Presutti V., Vitali F. "WikiFactory: an ontology-based application to deploy domain-oriented wikis". To appear in the *Proceedings of the European Semantic Web Conference 2006*, June, 2006, Budva, Montenegro.
6. FreeMarker, "FreeMarker templating engine", http://freemarker.sourceforge.net/index.html
7. Guzdial, M. Rick, J. and Kehoe, C.: "Beyond Adoption to Invention: Teacher-Created Collaborative Activities in Higher Education", *Journal of the Learning Sciences*, 2001, Vol. 10, No. 3, 265-279.
8. HP Labs, "Jena – A Semantic Web Framework for Java", http://jena.sourceforge.net/.
9. Jspwiki.org, WikiRPCInterface, http://www.jspwiki.org/Wiki.jsp?page=WikiRPCInterface
10. Krotzch Markus, Denny Vrandecic, and Max Volkel. "Wikipedia and the Semantic Web The Missing Links". In *Proceedings of Wikimania 2005*, Frankfurt, Germany, August 2005.
11. MediaWiki.org, "MediaWiki", http://www.mediawiki.org/wiki/MediaWiki.
12. Palmer Sean B. "RDFwiki". http://infomesh.net/2001/rdfwiki/.

13. Protégé, “The Protégé Ontology Editor and Knowledge Acquisition System”. http://protege.stanford.edu.
14. Souzis A. “Rhizome position paper”. In Proceedings of the *1st Workshop on Friend of a Friend, Social Networking and the Semantic Web*, September 2004.
15. Thoeny P. TWiki: Enterprise Collaboration Platform. http://twiki.org.
16. Thoeny P., “TWiki Success Stories”, http://twiki.org/cgi-bin/view/Main/TWikiSuccessStories.
17. Wikipedia.org, “Semantic Wikis”, http://en.wikipedia.org/wiki/Semantic_Wiki.
18. Wikipedia.org. Wikipedia Home Page. http://www.wikipedia.org.