WikiFactory: An Ontology-Based Application for Creating Domain-Oriented Wikis

Angelo Di Iorio, Valentina Presutti, and Fabio Vitali

University of Bologna, Mura Anteo Zamboni 7, 40127 Italy

Abstract. Wikis play a leading role among the web publishing environments, being collaborative tools used for fast and easy writing and sharing of content. Although powerful and widely used, wikis do not support users in the aided generation of content specific for a given domain but they still require manual, time-consuming and error-prone interventions. On the other hand, semantic portals support users in browsing, searching and managing content related to a given domain, by exploiting ontologies. In this paper we propose a specific application of web ontologies, applied to the wikis: exploiting an ontological description of a domain in order to deploy a customized wiki for that specific domain. We describe the design of an ontology-based framework, named WikiFactory, that aids users to automatically generate a complex and complete wiki website related to a specific area of interest with few efforts. In order to show the applicability of our framework, we present a specific case study that describes the main WikiFactory capabilities in constructing the wiki website for a Computer Science Department in a University.

1 Introduction

Wikis are collaborative tools used for fast and easy writing and sharing of content on the Web. They provide a simple, quick, informal way to create web sites, web applications, shared environment for discussion and document collections, tools for distributed cooperative writing, and so on. The success of grassroot, yet authoritative, information sources based on wiki technologies (examples include sites such as Wikipedia [25], Portland Pattern Repository [6] or World66 [28]) is greatly due to the informal, unimposing, encouraging relaxed attitude towards open contributions that is the true mark of wiki applications.

Wikis are, at heart, a collection of text documents that are displayed as HTML pages through a simple server-side collection of scripts. Interfaces for editing and creating new content, as well as text conventions for expressing some text styles and hypertext links, allow some complexity in the final result. Yet wikis remain fundamentally generic tools, and what they are good for, substantially, is creating large and flat collections of web pages linked to each other. It is very hard (or, rather, it is very easy but completely manual) to create structures, especially repeating structures, and preorganizing content and pages for large scale, structured, systematic expression of organized content.

Thus, while it might be very fascinating that users are allowed to create fairly sophisticated content and sites without special applications, and without

Y. Sure and J. Domingue (Eds.): ESWC 2006, LNCS 4011, pp. 664–678, 2006.

any knowledge of HTML, CSS, HTTP, and any other basic web technologies, it could be frustrating to discover that all wikis will require that not only the actual content pages, but navigation lists, intermediate pages, recurring substructures, etc., need to be specified manually and repeated manually as many times as required by the complexity of the content to be put on line. For instance, a university faculty that decides to put on-line a wiki to describe its course offering, might find appealing that the wiki allows easy editing of each course page to each individual professor, but also discouraging that it has to create one by one all individual pages for courses, classes, professors, rooms, events, exams, and so on.

Each domain suggests a basic and natural structure of subsites, navigation patterns, organization of home and intermediate pages, automatic inter-site links, down to the content template of each individual page. Current generation wiki clones do not provide tools for the aided generation of domain-oriented wikis. Furthermore, since a wiki is a live entity continuously generated and updated and modified, the need for aided generation of substructures exists throughout the useful life of the wiki itself.

The creation of domain-oriented publishing environments can be supported by the emerging Semantic Web technologies. The Semantic Web is an extension of the current Web where information is given in a machine understandable form [3] by means of a fundamental "'tool"': the ontology. Web ontologies are collections of semantic classes and relations that provide us with a powerful way to express assertions and constraints about the world, and consequently about every specific domain of knowledge. Some technologies have been introduced to manage such web ontologies: Web Ontology Language (OWL) [26], that is the standard language for the definition of Web ontologies, and Resource Description Framework (RDF) [27], that is the base model of the Semantic Web.

In this paper we propose a specific application of web ontologies, applied to the wikis: exploiting an ontological description of a domain in order to deliver a customized wiki for that specific domain. We have designed a framework, named WikiFactory, that automatically builds a domain-based wiki, taking in input the description of the world where that wiki will be used. The most relevant advantage of such approach is clear: simplifying the creation of the wiki pages, avoiding users to manually set their internal structures and links. Moreover, these pages are natively decorated with metadata, directly extracted from the input ontology. As semantic portals do, the domain-oriented wikis simplify the creation, searching and management of content in a specific domain of interest.

Actually, WikiFactory has another relevant goal, that is the delivery of a reliable, available and scalable wiki. Another limitation with current generation wikis, in fact, is that they work in a best effort way with no expectation of these requirements. Organizations who need strong guarantees in terms of high availability of the services, or guaranteed support for a large number of users at the same time, or for robustness of the service in face of hardware or network failures, would surely decide that no wiki can provide these guarantees, and therefore turn to different kinds of software. Thus, besides being domain-oriented, wikis

produced by WikiFactory are Qos-enabled, that is able to guarantee a given quality of service. We term these kind of wikis B-Wiki.

In this paper we only focus the attention on the ontological aspects of WikiFactory, by investigating some issues related to the use of ontologies for the creation of specialized wikis. Issues and solutions about QoS requirements and the deployment of the complete B-Wiki will not be discussed here. The rest of the paper is structured as follows. Section 2 discusses some related works. Section 3 describes the proposed WikiFactory framework with its principal components. Section 4 presents a case study that shows the applicability of the framework. Section 5 presents some concluding remarks and future works.

2 Related Works

All wiki applications (the so-called WikiClones, to stress that they are derived from the original and true WikiWikiWeb application [24]) share the same basic philosophy of open editing and a common approach to simple text-based syntax for editing and content writing, but distinguish from each other for the additional services and application modules they offer to their users. These features cover the most varied areas: PurpleWiki [11] provides a full control on content fragments, JotSpot[9] integrate collaborative tools, SnipSnap[10] allows users to in-line organigrams and UML diagrams and a lot of other examples can be cited.

Particularly interesting are those projects that aim at integrating wikis with Semantic Web technologies, known as "'SemanticWikiWikiWebs [23]"'. Pytypus[19] is a collaborative semantic engine that uses RDF as a base technology: wiki pages are annoted with RDF and stored in a semantic database, in order to be easily searched by agents. Platypus Wiki [17] offers a simple user interface to create wiki pages with metadata based on W3C standards such as RDF [27] and OWL [26]. Rhizome [2] allows users to create content with explicit semantics, with little effort. Instead of the common wiki syntax, Rhyzome uses a specialized plain-text format, called ZML: users can annotate documents and express statements on resources without using RDF, but simply editing ZML pages and following simple rules (server-side, a specialized engine transforms this content in RDF statements and manages a rich network of metadata). Similarly, RDFWiki [16] provides users a simple text-based interface to edit content but stores all data as RDF. Users can express predicates on objects, and all wiki content can be exported as N-Triples. Similarly PeriPeri[4] allows users to decorate the wiki pages, adding metadata to the top of a page like an email header. With WikSar[7], authors can also embed on pages commands to gather content from all the available data in the repository. SemperWiki[12] aims at integrating such searching and indexing funcionalities with a usable interface and a personal space for each user.

Other projects deal with the automatic extraction of information and reasoning about wikis. In [14] authors propose an implementation to make content of Wikipedia understandable and processable by machines. They suggest the introduction of typed links among pages as a simple way to achieve this goal,

without loosing the soundness and usability of Wikipedia. WikiSense [13] aims at applying data-mining techniques to MediaWiki (the wiki-clone which Wikipedia relies on) in order to automatically extract the underlying web of concepts and relations.

Even if not directly connected with wikis, many projects by the Semantic Portals Community share issues and solutions with our work and are worth to be mentioned and compared each other. For instance, OntoWebber [29] is an integrated system that exploits ontologies to create different models and views of data, and present them on the web. The declarative model behind OntoWebber supports users throughout the whole life-cycle of a web-site and allows them to easily instantiate web pages from heterogeneous data sources.

Similarly SCWP [21] uses ontologies to generate and manage community web portals and to make heterogeneous information easily sharable among heterogeneous sub-systems. The same research group produced KA2 [20], a flexible ontology that can be used for different domains and handled by different semantic tools. Based on an artificial intelligence approach, SEAL [1] exploits ontologies to strengthen searching, import, conversion and managing of data within web portals too. Particularly interesting is the approach proposed in EnerSearch [22], whose sub-goal is hiding the complexity of ontologies and providing users a simplified interface to search and insert data.

WikiFactory adopts a similar approach but designed and implemented for wikis: hiding the complexity within the system and simplifying the internal data model, it simplifies the automatic generation and management of content, and consequently indexing and searching.

3 Using Ontologies to Build a Domain-Oriented Wiki with WikiFactory

Building a domain-oriented publishing environment (for instance, a wiki) from an ontological description is not a simple and straightforward process. Two separate worlds, in fact, need to be merged into the output: on the one hand, we have the *domain*, a set of actors, properties and actions strictly related to the reality we are describing and apparently unrelated to the world of the wikis; on the other hand, we have the *wikis*, general tools for writing and sharing content on the web, that can be used for any domain. In order to present how WikiFactory works we will explain, firstly, what makes a domain-oriented wiki different from a generic one (and so, we will outline the features of an ontological description of such wiki) and then, how WikiFactory actually transforms such ontology into the final output.

3.1 Describing a Domain-Oriented Wiki

At first glance, a "'domain-oriented"' wiki can be defined as a wiki whose pages hold data and information useful in that domain. For instance, in a wiki customized for a university we expect that any home-page of a lecturer holds his

name, his title, office hours, a list of the courses he teaches and so on, while an enrollment page contains some instructions and a form. In a wiki of a football team we expect that there's a page about the league match calendar (formatted as a set of tables, one for each match-day), a page about the history (consisting of different paragraphs), a page with the short-list of the team, and so on.

A domain-oriented wiki is also characterized by a set of pages, linked each other according to some patterns. We expect, for instance, that any wiki about a football team has a home-page linked with a page about the short-list of a team, a page about the history of the club, and a page for any player, and so on. In a wiki for the University, we expect that, for any course, there is a home-page, a page with a list of exams (linked to another page for the exams enrollment) and so on. At this level, the wiki is not only composed by some disconnected pages (whose internal structures and data are described by the ontology), but contains well-defined sub-areas, composed of a proper set of linked pages.

The data mentioned so far cover only the first installation of a wiki. Although this automatic deployment saves users' time and resources, another issues still remains unsolved: updating wikis once they live on web servers. Today users have to manually perform these tasks: whenever a university that uses a wiki plans to provide a new course, for instance, an user manually adds to the wiki a home-page for that course, creates specific pages for the exams and manually links all these resources. It should be useful to use a wiki that automatically creates (or proposes to create) some pages, whenever other pages are manually created by the authors: in the example of a university course, we expect that all the pages related to a course are automatically added to the wiki, whenever an user creates a course home-page.

3.2 Deploying a Domain-Oriented Wiki

In the previous section we have identified three main aspects that a domain-oriented ontology for wikis can describe: static pages, clusters of pages and run-time behaviour. The main goal of WikiFactory is just translating these data into the actual wiki pages and scripts, following the same schema: (i) producing pages for each element of the domain, composed by a set of internal and structured components, (ii) producing clusters of pages organized according to a given pattern and (iii) adding to the wiki some scripts that generate clusters of pages or provide services at run-time.

Before going on discussing about the ontology within WikiFactory and its internal processing, it is worth giving an overview of the whole framework. The WikiFactory framework consists of four principal components, namely the *Ontology creation supporting tool*, the *Repository*, the *WikiFactory Application*, and the *QoS Manager* as depicted in Fig 1.

The framework uses two ontologies: the domain specific ontology, which describes the domain, and the WikiFactory ontology, which describes the generic wiki elements and contructs. These two input objects are elaborated by a designer with an ontology creation tool (e.g., Protege [18]), in order to produce an extension of the domain ontology, which describes services and structures of a

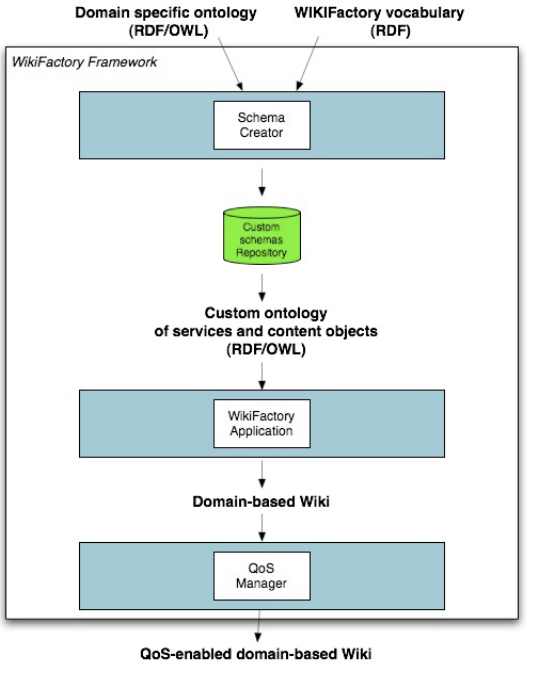

Fig. 1. The Wiki Factory Framework

wiki useful for that domain. Note that a manual intervention by the designer is required in order to translate the generic constructs of the WikiFactory ontology into specialized constructs for a given domain.

After having derived the domain-oriented wiki ontology, the designer simply saves it into the Repository. Any ontology extension stored in the database (one for each supported domain) can be processed by the WikiFactory Application. WikiFactory application is the core of the whole framework, in charge of translating ontologies into wikis: more details about this component will be provided in 4.2. The output of WikiFactory Application is then passed to the QoS Manager in order that produces and instantiates the final wiki (or rather, B-Wiki). As we said, discussing about the QoS Manager is out of the scope of this paper: here we only describe all the components involved in the deployment of a domain-oriented wiki, by focusing on their internal architecture and interaction.

We have identified two different users (i.e., roles) who use the WikiFactory for different purposes. Note that we do not want to define a methodology associated to the use of WikiFactory, but we believe that the description of these roles can be useful to explain how the framework actually works. The identified roles are:

– the final user, we name Bianca, i.e., an inexperienced user who adds content to the domain-oriented wiki and uses the final wiki every day for carrying out her tasks;

– an expert ontology designer, we name Andrea, who analyzes the requirements of the domain expert users in order to produce an extension of the ontology for that domain;

In order to explain how users (Bianca and Andrea) use WikiFactory, to provide readers more details about the role of each internal component, and to show a concrete application of our framework, we present a case study in the next section.

Actually, many other examples of wikis deployable from an ontological description could be also mentioned. Consider for instance, a wiki supporting Public Administration tasks: given a detailed description of roles, bureaucratic steps and forms, WikiFactory can create predefined areas and services, which in turn support and guide employees through their activities on the wiki itself.

Most wikis have also been growing and developing thanks to spontaneous contribution of different users on a shared interest. In that case, an ontological approach can be exploited to build structured pages (and clusters of them), that can be authored afterwards by the users, or partially authored by an automatic process. A wiki about travel information, for example, is supposed to have some pages about the amenities, the history, the facilities, the transportation system and whatever, for each reviewed destination: the structure of such a wiki can be derived from ontological data about travelling and cities. Similarly, a wiki for a community of wine lovers can be organized in different sections and subsections, according to the ontological classification of the reviewed wines, but thousands of similar examples can be cited.

4 A Case Study: Computer Science Department

The environment of our case study is the Computer Science Department (CSD) of a university, say the University of Bologna, that manages such common university activities as teaching, research projects and locations management, and bureaucracy. The CSD can use a wiki to make content and services accessible through the Web, and to support its internal workflows. This wiki is not a generic wiki but it has internal structures, content and services specific for the University domain, that can be described by an ontology. According to our approach, the CSD domain is described by an OWL ontology named CSD-Ontology. Fig. 2 shows the use case diagram drawn by Andrea for the CSD. Notice that, these use cases do not cover all the services the CSD needs and provides the users with; however, for the purpose of our discussion we can omit some of them.

Andrea has identified three possible actors in the CSD domain: Faculty, Students, and Administrative clerks. A Faculty member uses the wiki in order to reserve classrooms and/or laboratories for lessons and exams, to take a book from the library and order new books, to upload training aids, and so on. A student can use the wiki to browse and download training aids, to enroll for an exam, and to take a book from the library. The Administrative clerk uses the wiki in order to manage funds (for example a financial plan for a specific

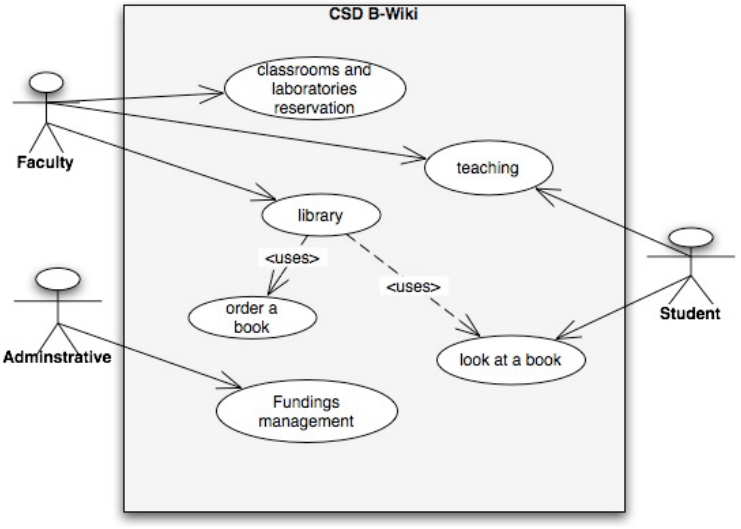

Fig. 2. The CSD Use case

research project). Let us show how such a wiki can be produced and deployed by WikiFactory.

4.1 Creating Ontologies

Any content and service mentioned so far, is delivered or implemented by a specific service of the CSD wiki. To this end, Andrea has to describe the CSD-Ontology extension for these custom wiki-services. Consider, for instance, the concepts depicted by the fragment of the CSD-Ontology shown in Fig. 3: a Professor holds a Course that is related to a specific Topic, that is, a Professor

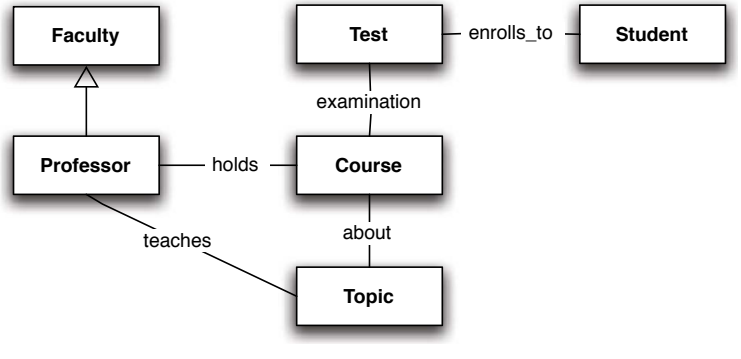

Fig. 3. The CSD-Ontology (a fragment)

teaches that Topic. A Student enrolls for a Test that is an examination for a Course.

A wiki for this scenario is supposed to have, first of all, a page for each Professor. That page is linked to other pages, one for each course thaught by the Professor (the ontology reports this relation, as well as the connection between a course and its subject). Furthermore, the wiki has some pages for managing each course, as a syllabus or a page to enroll students to the exam. Note that also this information can be partially derived from the ontology.

Andrea describes what such a wiki will consist of (which are the pages, how they are organized, which extra services are provided) and, subsequently, WikiFactory instantiates such a wiki by processing data inserted by Bianca (the final user). Basically, the task performed by Andrea is merging the CSD-Ontology concepts (provided him by domain experts) and the WikiFactory Ontology (already included in WikiFactory) into the CSD-Ontology extension. The ontology produced for the example, would just describe the above mentioned structure of a wiki, with pages for professors, courses and exams. As expected such an ontology follows the schema discussed in 3.1 and describes (i) the internal components of each page, (ii) the clusters of inter-connected pages and (iii) the dynamic behaviour of the wiki, when installed.

The following list briefly summarized a subset of the WikiFactory ontology concepts. Specifically the list gives an idea of the ontology and covers the concepts we need in order to describe the CSD scenario. First of all we need constructs to describe single pages:

- Topic Component: an element that composes a topic or topic template such as TextBlock, List or Table;
- Template: a set of topic components, organized according to a given structure;
- Topic: represents a page of the wiki; it consists of topic components and may or may not be associated to a template;

In order to describe clusters of pages, we need constructs to express relations and groups linked resources:

- Structure: a graph where each node is a topic and can be itself the root of a structure. It is used to model document hierarchies;
- Link: a piece of text associated to a URL address and that gives users the access to that resource;

Finally we need constructs to describe wiki services, behavior and actions. Although we are still discussing about this side of the ontology, we probably need only a construct that gives an high-level description of a service, linked with a resource that actually describes the service from a procedural perspective:

- Service: is a task performed by the wiki. The service has a reference to a resource that describes it in terms of a process.

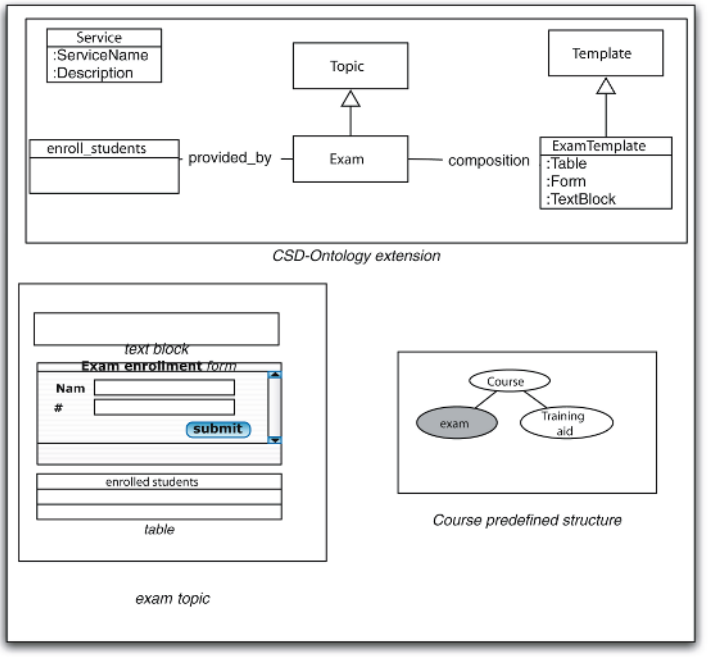

Fig. 4. The exam wiki topic

By referring to the above described fragment of the CSD-Ontology, Andrea derives the description of:

- the structure of a course: that is a predefined graph of topics (i.e., wiki pages) useful to manage content and resources related to a course (e.g. training aid, program and test enrollment);
- each topic in the structure;
- the exam enrollment service for the students.

Fig. 4 depicts the Course structure and the Exam topic definition. The structure Course represents a predefined graph of Topics related to the concept of course (when a Faculty uses the wiki he simply has to ask a new course area, and the related wiki pages are automatically created). The course structure consists of three topics: Course (the root of the structure), Exam, Training Aid. Andrea can also define constraints on the topics: for instance, he can state that the Exam topic can only exist in the context of the course structure (this means that the Exam topic cannot exist autonomously).

Moreover, Andrea can describe what each Topic consists of. For example, Exam is a topic that contains a text area, a form element, and a table element. The key aspect is that a relatively small set of Topic Component exists, that can be assembled to produce any final Topic. As we discuss in the following subsection, WikiFactory is able to produce a wiki, where any kind of page keeps

the structure expressed in the ontological description. In our case study, the CSD-Ontology says that the Exam entity consists of some specific elements, so any Exam page into the domain-oriented wiki has these elements.

The exam Topic is also associated to some services, i.e. operations to be performed within this topic. For instance, the exam enrollment is a service associated to the form and the table composing the topic Exam. What usually happens is summarized as follows: a student wants to enroll to an exam, he/she fills the form and the system automatically update the table. Andrea does not describe how the system works to provide this service, but he has to simply indicate a resource which contains this workflow description. In conclusion, he produces a description of all the interesting pages into a wiki for the University and all the services that can be useful in the same context. This description is stored in the repository, so that any University can produce its own domain-oriented wiki by giving in input the same RDF description to the WikiFactoryApplication.

4.2 Deploying Wikis with Wiki Factory Application

The WikiFactory Application, as shown in Figure 5, is the core engine of the system and is in charge of producing the domain-oriented wiki. Note that at this stage the produced output is not the final B-Wiki yet (domain-oriented and Qos-enabled); rather the output is a domain-oriented wiki with no QoS capabilities.

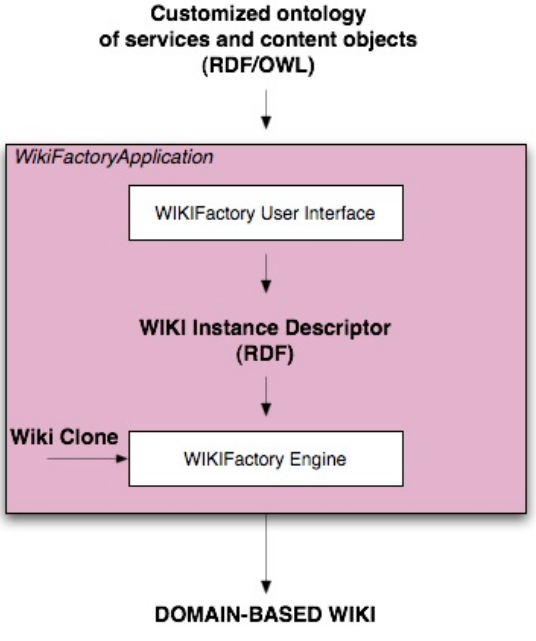

Fig. 5. The Wiki Factory Application

The actor involved in using the WikiFactory Application is Bianca, who wants to configure a wiki: from her perspective, the WikiFactory Application is a configuration tool that is being used in order to select the services the wiki will provide users and the topics it will consist of. Bianca, without having any skill about ontologies and the functioning of the wiki application, selects which wiki topics and services will be included in the final wiki installation. The internal process within the WikiFactory Application is completely transparent to Bianca, who has to simply fill in checkboxes, options and forms by means of a graphical interface.

She is first asked to select the domain, and then to choose the data to be included in the domain-oriented wiki. For instance, she indicates the name of lectures of the Department (and insert data for each of them) and the course they teach. Since the WikiFactory Application knows (from the ontology provided by Andrea) that any course has an exam, the final wiki will automatically have a page for each teacher, subsequently a page for each of his courses, and finally a page to handle the corresponding exam. Furthermore Bianca can select other services useful for the community such as forum, bullettin-boards, calendars.

What Bianca perceives as a simple task of selecting services, structures and topics and filling them with relevant data, is actually supported by a complex process within the WikiFactory Application. This aspect is worth being remarked: our framework is designed to minimize the effort required to the users, by hiding the complexity of the internal system components. The interface used by Bianca is not only usable but, most of all, it is customized for a specific domain: she does not have to learn a different formalism to express content and relations; neither has she to handle ontologies and wikis. Indeed, the WikiFactory Application, by taking in input the ontological description provided by Andrea, will dynamically present her a simple and transparent interface.

Note that our approach does not damage the easy-to-use approach of wikis, rather it does strenghten its power. The final output of WikiFactory, in fact, remains a wiki where users can keep on accessing and taking advantages from such systems (actually, being the application completely independent from the platform where content will be deployed, users can keep on using their preferred softwares too). What really changes is the process of authoring repeatable fragments, pages and clusters of pages which becomes simpler and faster, since some manual, error-prone and time-consuming actions are replaced by a simple selection of features and insertion of data.

In order to be processed within the system, the data inserted by Bianca have to be transformed into something closer to the wiki concepts (she does not describe them in terms of wiki objects, but as information). An intermediary output of the whole process is a Wiki Instance Descriptor. The Wiki Instance Descriptor is a description of what Bianca has selected and filled with data. There is a difference between a Wiki Instance Descriptor produced by Bianca and the extended ontology produced by Andrea: while the latter describes all the elements and services available in a given domain, the first describes only content

and services selected by the Department of Computer Science. Another user like Bianca working in a different department inserts data of different teachers and exams and probably selects different services. The RDF description stored in the Repository specifies what a wiki for the University can contain, while the Wiki Instance Descriptor says what the wiki of the Department of Computer Science actually contains.

In other words, the Wiki Instance Descriptor gives a high-level description of the outgoing wiki. The final step performed by the WikiFactory Application, in particular by a sub-component we termed WikiFactory Engine, is to map such a description into a specific wiki-clone. WikiFactory Application, in fact, does not produce a new wiki-clone, but customizes some of the existing wiki-clones in order to provide the same functionalities. The goal is not to produce another brother of TWiki [15] or Purple [11] UseMod [8], but to use (if necessary, by extending) each of them. In our case study, after having inserted data, Bianca is asked to select a specific wiki-clone on which the outcoming domain-oriented wiki will be installed.

The solution we propose relies on a strong assumption: it is possible to identify a set of basic wiki services and content elements available in any wiki-clone that compose the final pages. Any wiki-page, in fact, can be segmented into a number of objects like paragraphs, tables, lists and so on, although any clone uses its own syntax and constructs pages can be easily generated from an high-level description. Also the services can be combined in order to obtain more complex services and features. By having an abstract definition of each basic service and by knowing how they can be combined into a more complex one, the WikiFactory Engine actually transforms a Wiki Instance Descriptor into a wiki instance. We are investigating rules and patterns to decompose wiki services and a language to be used.

5 Concluding Remarks and Future Works

Wikis are a new and exciting technology whose applications are wide and extremely innovative. On the other hand, before organizations can reliably start using them for large scale applications, wikis need to improve on the limitations that the current implementations suffer from.

We have identified two key issues in wikis: on the one hand, support for automatic generation and maintenance of domain-oriented content and structures, and on the other hand specification of quality of service parameters that are honored by wiki applications to provide guarantees of scalability and availability of the provided services and content. In this paper we have investigated the automatic deployment of a domain-oriented wiki taking in input ontological description of the domain. The development of web applications according to ontologies, with examples taken from the description of a university department, has been already described in [5]. Moving off such analysis, we have designed a modular framework based on Semantic Web technologies that aims at creating a semantic wiki, minimizing users' effort.

Preliminary implementations of many of the described modules exist and have been tested independently but we are currently investigating on the integrability of these intermediate results.

WikiFactory is a lively ongoing project, whose content and results have only been sketched in this paper. More detailed and up-to-date results can be always found in its web site, justifiably enough a wiki itself, at the address http://swe.web.cs.unibo.it/WikiFactory/.

Acknowledgements

We wish to thank our colleagues Giorgia Lodi and Andrea Ceccanti who work on the Quality of Service aspects of WikiFactory and help us with interesting and fruitful discussions about the whole project. We also wish to thank the Prof. Fabio Panzieri of University of Bologna and our colleague Jaksa Vuckovic for their precious comments and suggestions on earlier versions of this paper.

References

1. Maedche A., Staab S., Stojanovic N., Studer R., and Sure Y. A Framework for Developing SEmantic portALs. In *18th British National Conference on Databases*, Oxford, UK, July 2001. LNCS Springer Verlag.
2. Souzis A. Rhizome position paper. In *Proceedings of the 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web*, September 2004.
3. Berners-Lee T., Hendler J., and Lassila O. The Semantic Web. *The scientific american*, 2001.
4. Chris Purcell. Periperi. http://www.srcf.ucam.org/ cjp39/Peri/PeriPeri.
5. P. Ciancarini and V. Presutti. Towards Ontology Driven Software Design. In M. Wirsing, S. Balsamo, and A. Knapp, editors, *Proc. 8th "Monterey Workshop": Radical Innovations of Software and Systems Engineering in the Future*, pages 158–168, Venice, Italy, October 2002.
6. Portland community. Portland Pattern Repository's Wiki. http://www.c2.com/cgi/wiki?WelcomeVisitors.
7. Aumueller David. Semantic authoring and retrieval within a Wiki. In *Demos and Posters of the 2nd European Semantic Web Conference (ESWC 2005)*, Heraklion, Greece, May 2005.
8. Herman. Moin Moin Wiki. http://twistedmatrix.com/users/jh.twistd/moin/moin.cgi/.
9. JotSpot Inc. Jotspot beta: the application wiki. http://www.jotspot.com/.
10. Jugel Matthias L. and Schmidt Stephan J. Snipsnap: the easy weblog and wiki software. http://www.snipsnap.org/space/.
11. Kim E. E. Purplewiki. http://purplewiki.blueoxen.net/cgi-bin/wiki.pl.
12. Daniel Kinzler. SemperWiki: a semantic personal Wiki. In *The Semantic Desktop - Next Generation Personal Information Management and Collaboration Infrastructure at the International Semantic Web Conference*, Galway, Ireland, November 2005.
13. Daniel Kinzler. WikiSense Mining the Wiki. In *Proceedings of Wikimania 2005*, Frankfurt, Germany, August 2005.

14. Krotzch Markus, Denny Vrandecic, and Max Volkel. Wikipedia and the Semantic Web The Missing Links. In *Proceedings of Wikimania 2005*, Frankfurt, Germany, August 2005.
15. Thoeny P. TWiki: Enterprise Collaboration Platform. http://twiki.org.
16. Palmer Sean B. Rdfwiki. http://infomesh.net/2001/rdfwiki/.
17. Platypus Wiki. The Semantic Wiki Wiki Web. http://platypuswiki.sourceforge.net/.
18. The Protégé Ontology Editor and Knowledge Acquisition System. http://protege.stanford.edu.
19. Pytypus Home Page. http://www.pytypus.org/.
20. Benjamins V. R., Fensel D., Decker S., and Perez A.G. KA2: Building Ontologies for the Internet: a Mid Term Report. *International Journal of Human-Computer Studies*, (51):687–712, March 1999.
21. Staab S., Angele J., Decker S., Erdmann M., Hotho A., Maedche A., Schnurr H., Studer R., and Sure Y. Semantic Community Web Portals. In *Proceedings of the 9th International World Wide Web Conference*, pages 1–6, Amsterdam, May 2000. ACM.
22. Staab S. and Studer R. Ontology-based Content Management in a Virtual Organization. *Handbook on Ontologies*, pages 687–712, 2003.
23. Semantic Wiki Wiki Web. http://c2.com/cgi/wiki?SemanticWikiWikiWeb.
24. Wiki Wiki Web. Cunningham and Cunningham Inc. http://c2.com.
25. Wikipedia. Wikipedia Home Page. http://www.wikipedia.org.
26. World Wide Web Consortium. OWL Web Ontology Languagefamily of specifications. http://www.w3.org/2004/OWL/, 2004.
27. World Wide Web Consortium. RDF Resourcs Description Framework family of specifications. http://www.w3.org/RDF/, 2004.
28. World66. World66 home. http://www.world66.com/.
29. Jin Y., Xu S., and Decker S. Ontowebber: Model-driven ontology-based web site management. In *Proceedings of SWWS'01, The first Semantic Web Working Symposium*, California, USA, July 30 - August 1 2001.