Crowdsourcing semantic content: a model and two applications

Angelo Di Iorio[1], Alberto Musetti[1], Silvio Peroni[1], Fabio Vitali[1]
[1] Department of Computer Science, University of Bologna, Bologna, Italy
diiorio@cs.unibo.it, musetti@cs.unibo.it, speroni@cs.unibo.it, fabio@cs.unibo.it

Abstract. **While the original design of wikis was mainly focused on a completely open free-form text model, semantic wikis have since moved towards a more structured model for editing: users are driven to create ontological data in addition to text by using ad-hoc editing interfaces.**

This paper introduces OWiki, a framework for creating ontological content within *not-natively-semantic* wikis. Ontology-driven forms and templates are the key concepts of the system, that allows even inexpert users to create consistent semantic data with little effort.

Multiple and very different instances of OWiki are presented here. The expressive power and flexibility of OWiki proved to be the right trade-off to deploy the authoring environments for such very different domains, ensuring at the same time editing freedom and semantic data consistency.

Keywords: **authoring, community, interfaces, ontologies, templates.**

I. INTRODUCTION

The explosion of social software tools has changed the way most users access the World Wide Web. Even inexpert users can now publish their content with a few clicks and do not need to master complex technologies or to use professional tools. This content is primarily targeted to being consumed by human users, as in YouTube videos, Twitter messages, FaceBook posts and so on. The creation of *semantic* web content – available for automatic search, classification and reasoning – is much more difficult and time-consuming. Technical competencies and domain-specific knowledge, in fact, are still required in authors. The shift of the Web from a *human-understandable* to a *machine-understandable* platform, as envisioned by the Semantic Web community [1], is far from being complete.

The term “lowercase semantic web” [2] has been coined to indicate research efforts aiming at bridging the gap between simplified authoring and semantic web data. Such lowercase semantic web is not an alternative of the uppercase “Semantic Web”, but rather an intermediate step towards the same goal. While the Semantic Web aims at bringing full-fledged reasoning capabilities to intelligent software, the lowercase “semantic web” aims at encoding semantic data that can be accessed by everyday software and, above all, can be created by unsophisticated users.

Semantic wikis play a leading role in such a scenario. Semantic wikis are enhanced wikis that allow users to decorate pages with semantic data by using simplified interfaces and/or specialized wiki syntaxes. They provide users with sophisticated searching and analysis facilities, and maintain in full the original open editing philosophy of wikis in everything but the form in which the content is created.

Many semantic wikis, though, are essentially designed for scholars and experts in semantic technologies, and are still difficult to be used by average wiki contributors with no technical expertise. Forms are often used to mitigate this issue, providing an intuitive interface that is well known to most computer users. Unfortunately semantic forms as they exist currently do not guarantee the simplicity and ease of use that average users may expect. In most cases, in fact, these forms use generic text fields that are not fit to the domain the wiki is used for, or that only include a limited set of interface widgets.

In this paper we introduce a methodology and a tool to simplify the process of authoring semantic web data through wikis, and we present two very different environments where that tool was delivered. The overall approach relies on ontologies and allows even inexpert users to create semantic data with little effort. The tool, called OWiki, uses Semantic Web technologies to handle the wiki knowledge-base and MediaWiki forms and templates to deliver intuitive interfaces to the final users. Forms and infobox templates are automatically generated from ontological data, and are completely customizable to change the nature, structure and constraints of the form widgets.

The paper is structured as follows: Section II gives an overview of the main approaches to semantic wikis; the following one, Section III, presents the OWiki approach, focusing on its ontologies and their application in this context, while Section IV goes into the details of a use-case and the internals of the system are discussed in Section V. Two different instances of OWiki are presented in the last part of the paper, before drawing some conclusions: Section VI introduces BYOG, a collaborative environment for creating and delivering customized touristic guides, while Section VII presents ACUMEWiki, a shared platform for creating multi-disciplinary vocabularies and concept maps.

II. RELATED WORKS: SEMANTIC WIKIS AND ONTOLOGIES

The integration and interaction between ontologies and wikis is a hot research topic. Several approaches to semantic wikis have been developed to bring together the benefits of the free editing philosophy of wikis and ontological data. Semantic wikis can be organized into two main categories according to their connections with the ontologies: “wikis for ontologies” and “ontologies for wikis” [3].

In the first case, the wiki is used as a serialization of the ontology: each concept is mapped into a page and typed links are used to represent object properties. Such a model has been adopted by most semantic wikis. SemanticMediaWiki [4] is undoubtedly the most relevant one. It provides users with an intuitive syntax to embed semantics, i.e. RDF statements, within the markup of a page. SemanticMediaWiki allows users to freely edit the content without any limitation. The more the information is correctly encoded the more semantic data are available, but no constraint is imposed over the authoring process.

The difficulty in creating semantic content is still an open issue for SemanticMediaWiki and similar projects. Although the syntax is very simple, in fact, authors still have to learn some new markup and above all to manually write correct statements. SemanticForms [5] is an extension of SemanticMediaWiki that addresses such issue by allowing users to create semantic content via pre-defined forms. SemanticForms generates forms from templates – i.e. predefined structures dynamically filled by content and rendered as tables within MediaWiki articles - whose fragments and data have been previously typed. The generation process exploits an embedded mapping between each datatype and each type of field (radio-buttons, checkboxes, textareas, etc.). Users do not need to manually write statements anymore as they are only required to fill HTML forms. On the other hand, these forms have a fixed structure that cannot be customized and are still difficult to be mastered by the administrators of the wiki. Moreover the semantic data expressed in the template and the inline content of a page are still two distinct resources that might even be inconsistent.

The extensive usage of OWL, templates and forms makes the integration between wikis and ontologies a concrete possibility. In fact, some researchers have also proposed SMW-mOWL, a “meta-model extension of SemanticMediaWiki”. SMW-mOWL is a way to encode an entire OWL ontology into a wiki by exploiting “semantic templates”. Basically it consists of storing ontology elements as template instances: classes, anonymous classes, properties, restrictions, individuals are all represented as templates, with a direct mapping between SMW-mOWL and OWL-AS. SMW-mOWL improves the ontological expressiveness of SemanticMediaWiki. Moreover, such an approach makes it easier to edit ontologies within the wiki itself as templates can be edited using the form-based interfaces of the SemanticForms extension [5]. On the other hand, the SMW-mOWL metamodel is conceptually difficult and requires high expertise to be fully exploited.

The second category of semantic wikis, based on the principle of “ontologies for wikis”, includes all those wikis that are actually built on top of ontological foundations. The idea is to exploit ontologies to create and maintain consistent semantic data within a wiki so that sophisticated analysis, queries and classifications can be performed on its content.

IkeWiki [6] was one of the first wikis to adopt this approach. Its deployment starts by loading an OWL ontology into the system that is automatically translated into a set of wiki pages and typed links. Multiple interfaces are provided to the users for editing the plain wiki content, adding new metadata or tagging pages. IkeWiki strongly relies on Semantic Web technologies: it even includes a Jena OWL repository and a SPARQL engine used for navigation, queries and display of the semantic content of the wiki.

SweetWiki [3] implements a user-friendly ontology tool designed for both expert and non-expert users. The system is characterized by two aspects: the strong connection with the ontologies and the provision of Ajax-based interfaces for editing content and metadata. SweetWiki defines a “Wiki Object Model”, i.e. an ontology describing the wiki structure. Concepts like “document”, “page”, “link”, “version”, “attachment” are all codified in an OWL file that is accessed and manipulated through the wiki itself. These concepts are made explicit in SweetWiki, although they are usually hard-coded in most semantic wikis. SweetWiki also allows users to import external ontologies and, once again, to access and manipulate those ontologies through the wiki interfaces. Finally, the system includes a third ontology that is dynamically created by the users through “assisted social tagging”: users can add metadata to any page and can put pages in relation. These metadata values form a folksonomy that, on the one hand, is freely editable by users and, on the other, is built on top of ontological data. The interface for tagging, in fact, suggests consistent metadata by exploiting SPARQL queries and autocompletion features.

Finally, UFOWiki [7] is another project that aims at integrating wikis, ontologies and forms. UFOWiki is a wiki farm, i.e. a server that allows users to setup and deploy semantic wikis. Multiple wikis are deployed by the same farm, so that they can share (ontological) data in a distributed environment. The overall content is stored in a centralized repository as RDF triples that express both the actual content of each page and its metadata. Multiple ontologies are used to model these data. The SIOC ontology (http://sioc-project.org) is used to describe wiki pages and users' actions, helped by a domain ontology that can be imported and mapped into the wiki. Users are also provided a combination of plain-text editors and forms to modify the ontology within the wiki. The UFOWiki forms are generated on-the-fly starting from the mapping between the classes and properties of the ontology and the fields and types of the form. The wiki administrators are in charge of setting this mapping through a graphical and intuitive interface.

The high configurability of UFOWiki forms is one of the most relevant features, but also the ontological expressiveness is another one. In fact, UFOWiki allows users to create complex ontology instances and relations within a single page. While most other wikis only create assertions whose subject is represented by the subject of the wiki page containing that assertion, UFOWiki allows users to associate sub-forms to classes of the ontology and to handle these parts as separate resources. The result is a more fine-grained control over the ontology and its populating process.

III. OWIKI: ONTOLOGY-DRIVEN GENERATION OF TEMPLATES AND FORMS FOR SEMANTIC WIKIS

OWiki is Gaffe-based [8] extension of MediaWiki that supports users in creating and editing semantic data. The basic idea of OWiki is to exploit ontologies and MediaWiki editing/viewing facilities to simplify the process of authoring semantic wiki content.

In particular, OWiki exploits MediaWiki templates, infoboxes and forms. A *template* is set of pair *key-value*, edited as a record and usually formatted as a table in the final wiki page. Templates are particularly useful to store structured information: very easy to edit, disconnected from the final formatting of a page, very easy to search, and so on. Templates are defined in special pages that can be referenced from other pages. These pages include fragments with the same structure of the template but filled with instance data. The template-based component of a page is also called *infobox*.

OWiki exploits ontologies to represent the (semantic) knowledge-base of a wiki and templates to display and edit that ontology through the wiki itself. The integration and interaction between ontologies and templates can be summarized in two points:

- each class of the ontology is associated to a template-page. Each property is mapped into a key of the infobox;
- each instance of that class is represented by a page associated to that template. Each line in the infobox then contains the value of a property for that instance. Data properties are displayed as simple text while object properties are displayed as links to other pages (representing other instances of the ontology).

Currently OWiki does map *all* instances of the domain ontology into wiki pages and all properties into lines of the infoboxes. We are currently extending the engine in order to also allow users to select a subset of concepts to be represented by wiki pages, and a set of properties to be shown in those pages.

OWiki templates are actually transparent to users. In fact, each template is associated to a form that allows users to create and edit the relative instances. Users do not modify the templates *directly* but they only access specialized form fields.

The crucial point is that even forms are generated on-the-fly from ontological data. OWiki also includes a *GUI ontology* describing widgets and interface elements. The concepts and relations of the *domain ontology* – that is the ontology in which metadata content is specified – are mapped into form elements that are delivered to the final user.

During the installation phase OWiki creates a basic set of forms by merging the domain ontology with the GUI one. At the editing phase, the system shows a very basic form and saves it as a special page (template). This page can then be organized as a new form by adding dynamic behaviours, moving buttons, changing the field order and so on.

Before describing the internal architecture of the system, it is worth spending few more words about the way OWiki uses ontologies. In fact, the extensive usage of ontologies makes it possible (i) to make OWiki independent on the domain it is used for, (ii) to easily customize forms and templates, and (iii) to fully describe the evolution of a wiki page and its semantic content.

A. *Using ontologies to model the domain*

In OWiki the entire *domain of discourse* – i.e., all the topics each page talks about – is handled by using an OWL ontology, called *domain ontology*. Two different kinds of classes exist in this ontology: those – *page-domain* classes – strictly related to articles and pages visualized by the wiki, and other ones – *data-domain* classes – that define additional data around the former ones.

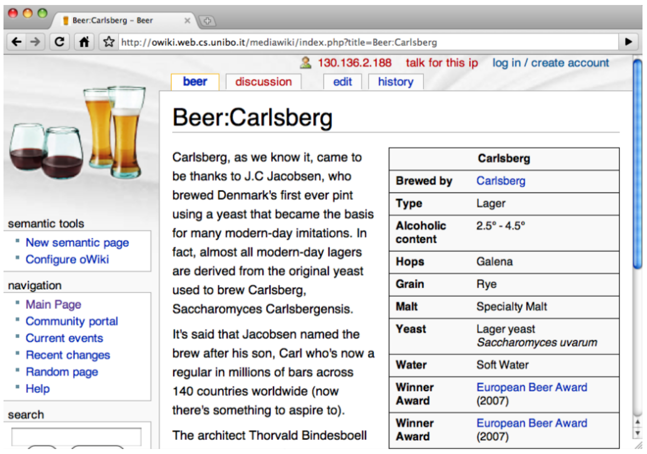
As we have previously said, each *page-domain* individual results in a wiki page containing text content (the content is now stored in the MediaWiki internal database) and all semantic data directly related to that individual. Fig. 1 shows a page about a particular beer (available at http://owiki.web.cs.unibo.it) that contains a text description of it in the central page area, while in the right box there are all metadata about this particular beer.

While some metadata, such as "Beer Alcoholic content" or "Beer Brewed by", are proper to any beer directly, because they are defined by OWL data or object properties having the class *Beer* as domain, that is not true for others metadata, such as "Winner Award" and "Winner Awarded on". In fact, those properties are handled using the *data-domain* class *Awarding* that represents an event, concerning a particular prize, in which a beer participated to in a specific time. The model behind the value of such properties for the beer shown in Fig. 1 is explained in the following excerpt (in Turtle syntax):

```
:carlsberg
 a :Beer ; :hasAwarding :awardingEuropean2007 .

:awardingEuropean2007 a :Awarding ;
    :hasAward :europeanBeerAward ;
    :hasYear “2007” .
```

The values shown in the Carlsberg page are not directly extracted from the Carlsberg ontological individual: they are taken from the awarding event the Carlsberg beer participated to.



Beer:Carlsberg

Carlsberg, as we know it, came to be thanks to J.C Jacobsen, who brewed Denmark's first ever pint using a yeast that became the basis for many modern-day imitations. In fact, almost all modern-day lagers are derived from the original yeast used to brew Carlsberg, Saccharomyces Carlsbergensis.

It's said that Jacobsen named the brew after his son, Carl who's now a regular in millions of bars across 140 countries worldwide (now there's something to aspire to).

The architect Thorvald Bindesboell

Carlsberg	
Brewed by	Carlsberg
Type	Lager
Alcoholic content	2.5° - 4.5°
Hops	Galena
Grain	Rye
Malt	Specialty Malt
Yeast	Lager yeast *Saccharomyces uvarum*
Water	Soft Water
Winner Award	European Beer Award (2007)
Winner Award	European Beer Award (2007)

Fig. 1. An example page of the Beer OWiki.

Even if they are not directly represented as wiki pages, OWiki uses *data-domain* individuals to enrich even more the metadata of *page-domain* individuals. This enrichment needs to retrieve related data from non-page-domain-individuals making at least two steps on the RDF graph represented by the model. We call *property flattening* the visualization of those *data-domain* property values into a *page-domain* individual.

B. *Using ontologies to model the interface*

OWiki exploits ontologies to also model end-user interfaces. In particular, the system includes a GUI ontology identifying all the components of web forms. The system instantiates and merges that ontology with the domain one in order to generate the final forms. The definitive version of the GUI ontology is still under development but the core concepts and relations are stable and already tested in the current prototype.

Separating the GUI ontology from the domain one has a two-fold goal: (1) generating a declarative description of the interface widgets that can be reused across multiple domains not being bounded to specific data and (2) allowing users to customize final interfaces by only changing the association between content and interface widgets. Note also that the GUI ontology can be designed once for all, while the domain one requires different expertise for different application scenarios.

The GUI ontology defines two types of graphical elements: *controllers* and *panels*. Panels are containers for other elements (that can be panels, in turn) used to organize the overall interface, while controllers are single widgets allowing users to actually fill metadata.

The main class of the ontology is *OWikiForm*. Instances of this class will be used to generate each form associated to each wiki page. Each instance will in fact contain either graphical elements or property values from the domain ontology. *OWikiForms* can contain *simple* or *complex* types of controllers. Simple types will be associated to data properties in the domain ontology, while complex type will be associated to object properties.

Simple types model the basic form elements (*Textfield*, *ComboBox*, *CheckBox* e *RadioButton*) while complex types model constructs useful for mapping the GUI ontology to the domain one. In fact, there are two complex types: *ConnectField* and *ObjectContainer*. *ConnectField* model links to another wiki document. This will be finally used to provide users with auto-completion operations on corresponding form fields: when the user will fill this field, the system will suggest a set of linked documents she/he can choose from (or create a link to a completely new resource). These links are in fact derived from the relations in the domain input ontology. *ObjectContainers* are widgets that includes properties of a class non-directly linked to the one defining a particular page, including into documents data about other (related) subjects. This class implement what we illustrated previously as *property flattening*.

IV. STUDYING OWIKI THROUGH A USE-CASE

The main goal of OWiki is to simplify the creation of semantic data *through and within* wikis. The complexity of such metadata authoring process, in fact, is hidden behind the application in order to not force users to learn new interfaces and tools. They can easily create semantic data by exploiting forms and templates that are automatically generated from ontological data.

In this section we explain with much details this generation process, clarifying how ontological data are converted into (customized) interfaces. Basically, the overall OWiki process consists of three steps:

1. ontology import and forms generation;
2. forms customization;
3. templates and data generation.

A. *From ontologies to forms*

The first step consists of importing the input domain ontology into the wiki. Let us consider a sample application we will discuss throughout the following sections: an OWiki demo installation describing beers, breweries, ingredients, etc. Fig. 2 shows some classes of a domain ontology suitable for such an application.

Classes and properties are mapped into wiki pages following the schema briefly described in the previous section: each concept is mapped into a page and properties are expressed through templates. In particular, data properties become lines of templates infoboxes and object properties becomes typed links.

Note that the overall status of the OWiki installation is consistent at this stage, assuming that domain input ontology was consistent. The process is in fact a straightforward translation of classes and relations into pages and links.

The OWiki conversion process also produces forms to edit the ontological content. Forms are dynamically built by analysing the class properties of the imported ontology and by mapping each property in the proper element of the GUI interface.

In the example, the class *Beer* defines three properties: *name*, *type* and *alcohol content*. According to the type of these properties OWiki generates text fields or radio buttons. The default element is a text field that allows any type of value. Since in the input ontology the only possible values of the property *beer type* are *Ale*, *Lager* and *Pilsner*, the system add to the form a RadioButton element specifying those values.

For object properties OWiki chooses between two types of widgets according to their range: whether the range class is a descendant of the *oWiki* class, the system adds a *ConnectField* to the form; otherwise it adds an *ObjectContainer*. Since the *Beer* class has the object property *brewed by* with the *Brewery* class specified as range, for example, the system add to the form a widget that allows to include a link to a corresponding brewery page. This widget will also provide auto-completion features built on top of the relations expressed in the input ontology.

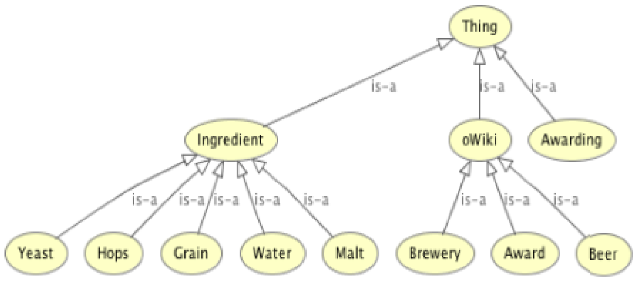


Fig. 2. A graphical representation of the OWiki domain ontology about beers.

A point is very important at this stage: there is a default mapping between classes of the domain ontology and elements in the GUI ontology based on *the type of the properties*. The name of a property or its meaning in a specific domain is not relevant.

There is actually a configuration file that specifies, for each type, which widget to use and how to configure it. In the previous case, for instance, there was an association between enumerations and radio buttons. That mapping is deployed whenever a class has a property which may only have a finite set of values, regardless of the actual domain ontology. In fact, a change in the OWiki configuration file would be reflected in using a different widget for the same property.

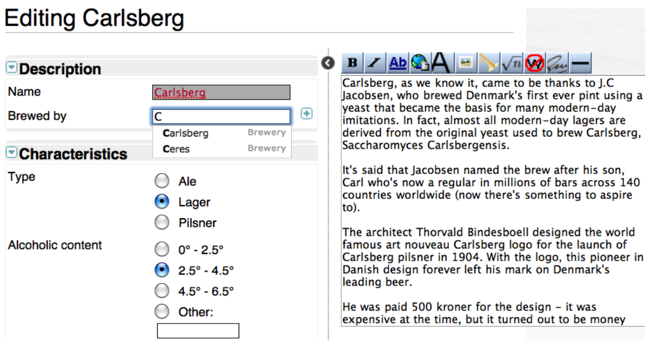
B. *Forms customization and filling*

Furthermore OWiki includes a configuration interface that allows users to set a domain-specific mapping between the input (domain and GUI) ontologies, and to configure the overall organization of the form and its formatting properties.

The first time a user edits a page, OWiki shows a basic form. The author can then organize a new form adding dynamic behaviours, moving buttons, changing fields order and so on. Fig. 3 shows a simple example of a customized forms: while the original form only listed a set of plain text-fields, this one is organized in panels and uses radio-buttons, images and dynamic widgets.

Customization can happen at different level. The user can change colour, font, background of the text to increase the appeal and impact of the form; she/he can change the position and the order of the elements to increase the importance of certain data; she/he can change the optionality of the elements, their default values, and so on.

The current implementation requires users to customize forms by editing an XML configuration file, through the wiki itself. Even if such an approach is not optimal, the internal architecture of the system relies on a strong distinction between the declarative description of the form (through the GUI ontology) and its actual delivery. That makes possible to implement a user-friendly and graphic environment to create and customize forms. One of the our future activities is the implementation of such an editor within the OWiki framework.


Editing Carlsberg

Description

Name	Carlsberg
Brewed by	C

Carlsberg Brewery
Ceres Brewery

Characteristics

Type	Ale [✓ Lager] Pilsner
Alcoholic content	0° - 2.5° [✓ 2.5° - 4.5°] 4.5° - 6.5° Other:

Carlsberg, as we know it, came to be thanks to J.C Jacobsen, who brewed Denmark's first ever pint using a yeast that became the basis for many modern-day imitations. In fact, almost all modern-day lagers are derived from the original yeast used to brew Carlsberg, Saccharomyces Carlsbergensis.

It's said that Jacobsen named the brew after his son, Carl who's now a regular in millions of bars across 140 countries worldwide (now there's something to aspire to).

The architect Thorvald Bindesboell designed the world famous art nouveau Carlsberg logo for the launch of Carlsberg pilsner in 1904. With the logo, this pioneer in Danish design forever left his mark on Denmark's leading beer.

He was paid 500 kroner for the design – it was expensive at the time, but it turned out to be money

Fig. 3. A customized form generated by OWiki.

C. *From semantic data to templates and views*

Automatically-generated forms are finally exploited by the wiki users to actually write the semantic data. As described in the previous section, data are stored as templates and templates are manipulated by forms *in a transparent manner*.

Let us consider again the *Beer* class of the example. OWiki generates a form to create instances of that classes showing three main components:

- a text field to insert the name of the beer
- a radio-button to select the type of the beer. Values in the radio button are directly extracted from the domain ontology.
- a text field to insert the brewery, that suggests breweries by exploiting information in the domain ontology.

These components can even be organized in multiple panels. Once the user fills the form OWiki saves a template with the proper information. Infobox templates, in fact, are used to display metadata and to cluster information about the same document.

Each infobox line correspond to a field in the form that, in turn, corresponds to a parameter and its value in the domain ontology. As expected, the data property of a class are displayed as simple text while the object property are displayed as links to other documents.

The page corresponding to the Carlberg beer in the example, that is an instance of the class *Beer* and has been edited via the corresponding form, will contain the following (partial) infobox:

```
{{Infobox Beer |
hasoWikiNamePage=Carlsberg |
Beer_brewedBy=[[Brewery:Carlsberg|Carlsberg]] |
Beer_beerType=Lager |
Beer_hasAlcoholicContent=2.5° - 4.5° |
Hops_hasName=Galena | … }}
```

Notice that the property *Beer_brewedBy* contains a link to the page *Carlsberg* that is now an instance of the *Brewery* class. Relations in the input ontology are then mapped into the links between pages. And the *Carlsberg* instance follows the same approach, being it described by the infobox:

```
{{Infobox Brewery |
hasoWikiNamePage=Carlsberg |
Brewery_hasAddress=Valby 11 DK - 2500,
 Copenhagen |
Brewery_brews=[[Beer:Carlsberg|Carlsberg]] }}
```

Some final considerations are worth about the consistency of OWiki. First of all, note that OWiki forms only work on the instances of the underlying ontology, without any impact on the classes and relations among them. The consequence is that, assuming that users do not corrupt infoboxes (that are anyway available in the source code of a wiki page), the overall ontology keeps being consistent. The OWiki instance is in fact consistent *by construction* with the domain and the GUI ontology and it is populated via forms in a controlled way.

Thus, we can conclude – going back to the distinction between "wikis for ontologies" and "ontologies for wikis" proposed in the related works section – that OWiki currently belongs to the second group and does not properly use the wiki to build and update ontologies. In the future we also plan to investigate a further integration between the wiki and the ontology – and a further integration between the textual content of a wiki page and the relative infoboxes – in order to also use OWiki as a full-fledged simplified authoring environment for ontologies.

V. THE ARCHITECTURE OF OWIKI

OWiki is an integrated framework composed of three modules, delivered with different technologies:

- a *MediaWiki extension*. It is a module integrated in MediaWiki written in PHP that adds OWiki facilities;
- an *Ontology manager*. It is a Java web service that processes OWL ontologies to produce forms for editing metadata. This manager uses internally both Jena API (http://jena.sourceforge.net) and OWLAPI (http://owlapi.sourceforge.net);
- an *Ajax-based interface*: a client-side module that allows users to actually insert data through the forms generated by the OWiki engine.

The PHP OWiki module follows the same architecture of any MediaWiki extension: some scripts and methods are overridden to provide new features. In particular, the module implements a revised editor that initializes OWiki environment variables, sets the communication with the client and sets data necessary to store forms in the MediaWiki database without interfering with existing data.

To manipulate ontologies, OWiki implements a web service that uses internally the Jena API. Jena is integrated with the Pellet reasoner (http://pellet.owldl.com), that is exploited to extract information about the instances in the ontology. Ranges of some properties, as well as their values, are in fact derived from subsumptions or other relations expressed in the ontology itself.

The web-service actually generates templates from the ontological data, that are later sent to the PHP module and stored in the right place of the MediaWiki installation.

The connection between the PHP and Java modules, and the core of the overall framework is the OWiki client. The client is a javascript application, based on Mootools (http://mootools.net), in charge of actually generating and delivering forms. It is strongly based on the Model-View-Controller (MVC) pattern and its internal architecture can be divided in four layers:

- *The Connection Layer* manages the overall environment, the initialization phase and the communication between all other layers.
- *The Model Layer* (Model of MVC) manages the data to be displayed on the page. It is composed of a factory that creates wrappers for each type of data and instantiates data from the ontology.
- *The LookAndFeel* (View of MVC) manages the final representation of the form, containing atomic and complex widgets, manipulators and decorators.
- *The Interaction Layer* (Controller of MVC) implements the logic of the application, the communication with the web-service, the generation of semantic data and the end-user interaction.

VI. CREATE AND DELIVER COMMUNITY-DRIVEN TOURISTIC GUIDES

Apart from the demo wiki about beers already mentioned, we have used OWiki to develop two instantiations of community-driven semantic wikis, BYOG (http://byog.web.cs.unibo.it) and the Epistemological Grid (for the European project ACUME2, http://acume2.web.cs.unibo.it), that will be illustrated in this section and in the following one.

BYOG (Book Your Own Guide) is a web application that aims at creating wikis to write, share and print customized touristic guides according to user needs. Its main features are:

- it uses crowdsourcing to write touristic guides, as well as content drawn from other sites;
- it allows users to freely mesh-up free and structured content in itineraries and full guides;
- it provides sophisticated social tagging mechanism;
- it uses print-on-demand technologies to print batches of guidebooks, and have them sent directly to user-specified addresses;
- it provides a free access to its content and also to a number of commercial services.

The three key concepts handled by BYOG are:

- **Point of Interest (PoI).** All text and metadata containers represent basic content entities. Each of them correspond to individual wiki pages. Each PoI correspond both to a particular wiki page and, as also shown in Fig. 4 to points on maps and they are identified by a flag icon (or something similar depending on the software used to visualize maps). Consequently, a PoI can have text, images, metadata and so on.
- **Itineraries.** Itineraries are PoI containers, represented by ordered lists of PoIs that can be named, described and stored also as wiki pages. Each itinerary is also a polygon on maps having PoIs as vertexes and they can be created by hand, by selecting individual PoIs, generated automatically as a result of a search on the database, etc. As PoIs, the itineraries can have text and metadata associated.
- **Guides.** A guide is an ordered list of itineraries (and schedules). Each guide contains the metadata needed to create a printed book (e.g., author, ISBN, price, etc.), apart from those specific to the guide itself. Whatever is applied to itineraries is applied to guides as well, with the exception that guides cannot directly contain PoIs. As PoIs and itineraries, guides also correspond to wiki pages.

These three concepts are semantically described as sub-classes of *owl:Thing* in a well-defined OWL ontology, that has the role of driving the overall content management process, according to the schema described in Section III.

This ontology is in fact composed by two parts: a BYOG-specific one that models PoIs, Itineraries and Guides and a wiki-specific one that models wiki articles and content. The relation between these two components is very strong. Instances of the ontology are in fact mapped into wiki articles, their properties are displayed in infoboxes and indirect properties are *flattened* as explained in Section IIIA.

Let us consider a very basic example: a user wants to create a guide with a single itinerary that includes some PoIs, each corresponding to a different hotel in the city of Bologna. The city of Bologna is in turn a PoI, previously included in the knowledge-base.

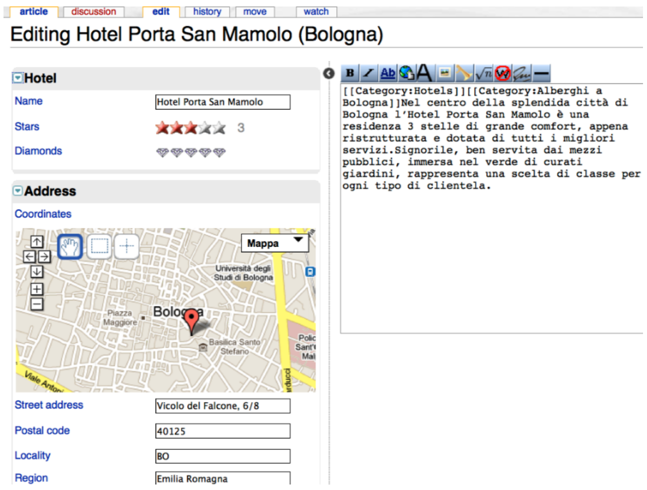
When the user creates the wiki-page about a new hotel – say “Hotel Porta San Mamolo” – two actions are actually performed: (i) a page is added to the wiki, editable by all other users and (ii) a new instance of the class *Hotel*, subclass of the class *PoI,* is added to the underlying ontology.

The wiki page is composed by two parts: a textual description of the PoI and an infobox that summarizes ontological data about that PoI and connections with other PoIs in the system. Similarly the page editor is composed by two parts: a textarea, where users can freely modify the page content, and a form, where users can add structured information about the hotel.

The form is generated automatically by OWiki, from the ontological description of the class Hotel. For instance, since the hotel is characterized by a name (a sequence of characters), a number of stars (an integer included from 1 to 5) and a location city (a link to another PoI), the form contains an input text (for the name), a graphical widget (for the hotel quality) and a select drop-down menu (to select the city where the hotel is located). Fig. 4 shows the editing interface for the PoI “Hotel Porta San Mamolo”.

BYOG includes the declarative descriptions of completely different forms to provide support for completely different PoIs such as museums, monuments, restaurants or hotels. These descriptions are built by combining the form widgets modelled in the GUI ontology with the data in the domain ontology (see Section III for more details).

When saving the page, the data inserted through the form are saved into the knowledge-base and shown in the infobox. Similarly to the beer example discussed earlier, BYOG users can customize the form (if they have access permissions to do so) and change both object properties (adding typed links to pages corresponding to PoIs or other objects in the ontology) or data properties (changing atomic values in the form).



Editing Hotel Porta San Mamolo (Bologna)

Hotel

Name	Hotel Porta San Mamolo
Stars	★★★☆☆ 3
Diamonds	

Address

Coordinates

Street address	Vicolo del Falcone, 6/8
Postal code	40125
Locality	BO
Region	Emilia Romagna

```
[[Category:Hotels]][[Category:Alberghi a
Bologna]]Nel centro della splendida città di
Bologna l'Hotel Porta San Mamolo è una
residenza 3 stelle di grande comfort, appena
ristrutturata e dotata di tutti i migliori
servizi.Signorile, ben servita dai mezzi
pubblici, immersa nel verde di curati
giardini, rappresenta una scelta di classe per
ogni tipo di clientela.
```

Fig. 4. Editing the PoI “Hotel Porta San Mamolo (Bologna)”.

Once collected all PoIs, the user can switch to the guides editor, that allows him to create detailed travel guides. This editor is a highly interactive Ajax-based application, very intuitive and easy to be used by inexpert users. The BYOG framework is actually completed by other modules that are out of the scope of this work: a semantic engine, to search for detailed content, an importer, to generate PoIs from external web-sites and a printing-on-demand module, for the actual creation and delivery of physical guides.

VII. USING OWIKI TO INTERFACE HETEROGENEOUS DOMAINS OF KNOWLEDGE

ACUME2 is an European Thematic Network Project (ETNP) within the Socrates-Erasmus Programme to interface sciences, literature and humanities. The intention of ACUME2 is to approach the concept of interfacing both from a truly theoretical as well as from an applied point of view, in order to build an infrastructure where researchers from various disciplines can communicate without ambiguities.

One of the problems encountered when collecting the output of very different communities and experts is the clashes in vocabularies and the different uses of event the same terms by different disciplines. To provide a guidance that does not prevent communities to use such terms, but at least confesses and expresses the differences in meaning, the main target of Acume2 has been the implementation of a *epistemological grid* for all the sub-projects of the thematic network. The grid is a table of definitions of terms created by the user, whose points of intersection are represented by the definition of a pair "term-value". In a community where several people have the task of defining common terms according to different disciplines it is useful to have a common model that is able to define better the meaning of terms.

It was soon realized that generating just a bunch of different definitions for the same terms would have provided very little insight on the real differences and similarities. For this reason it was established that comparing also how each term refers to others within the same discipline and outside would have been a much interesting way to bring light to similarities and differences. The aim of the epistemological grid is therefore to have a free-text description of the term for each discipline and each term, and in addition to generate a network of labelled references to other terms and concepts, again for each term and each discipline.

The ACUME2 epistemological grid has been developed and finally deployed through an instantiation of OWiki, called AcumeWiki. The system, in fact, managed to provide exactly the right kind of technological infrastructure needed to perform connections and relations among terms. The balance between the wiki free editing model and the aided production of ontologically consistent data was particularly appreciated by researchers. In fact, several researchers from 12 research institutes and universities collaborated to create the epistemological grid composed of dozens of terms and hundreds of relations.

Before going on discussing the deployment of AcumeWiki, it is necessary to briefly describe the conceptual models behind the epistemological grid. Two different conceptual models were considered and applied: the SKOS thesaurus, currently developed within the W3C framework (http://www.w3.org/TR/skos-reference), and the McLuhanian tetrad [9].

SKOS, or Simple Knowledge Organization System, is a family of formal languages designed for representation of thesauri, classification schemes, taxonomies, subject-heading systems, and any other type of structured controlled vocabulary. SKOS is built upon RDF and RDFS, and its main objective is to enable easy publication of controlled structured vocabularies for the Semantic Web.

SKOS defines a “concept scheme” that allows the use of mainly hierarchical relations (such as broader, narrower, and related) that are relevant in analysing and classifying a concept space. The concept scheme can then be locally extended with new concepts referring to existing ones, e.g. as specializations of these. Thus we can obtain connections between the definitions terms inside the same discipline. The use of SKOS within the epistemological grid, therefore, provides support for the correct placement of each term within a specific semantic tree, and sheds light on their closest related terms.

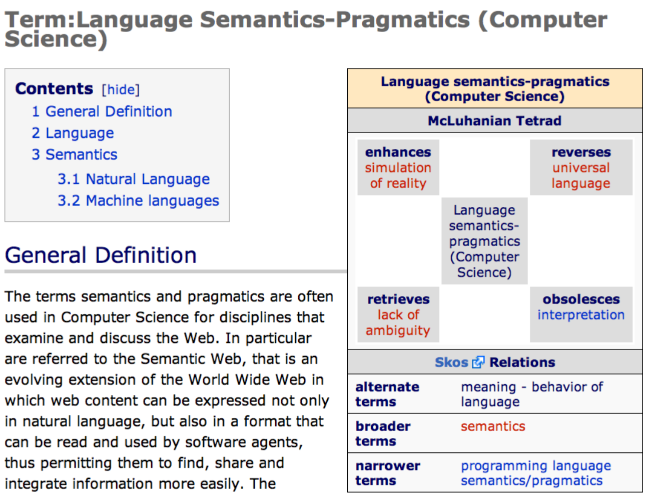
We were also suggested by our humanistic counterparts to adopt for term definitions the tetrad of effects originally devised by Marshall McLuhan for the effects of media [9]. McLuhan supplied a “definition scheme” that allows the expression of four specific relationships between the effects of media: *enhance*, *retrieves*, *reverses into* and *obsolesces*.

The complexity of the various types of tetrad is generated by the different kinds of people and then the different analysis of the same concept. This approach can give as result some insights and considerations that are suggestive of the multiple nature and complementarity of each definition of the same term.

Both SKOS and the McLuhan models were combined in AcumeWiki to improve the quality and richness of the epistemological grid. The OWiki domain-ontology (see Section IIIA for more details) was in fact composed by two parts: the imported SKOS ontology and the ontological representation of the MCLuhan model.

In particular, the ontology includes three main classes: a *Document* class corresponding to the wiki page, a *Term* class, corresponding to the term in the grid, and a *Concept* class, corresponding to the concepts in the tetrad. The class *Concept* is then further specialized in *Enhance*, *Retrieves*, *ReversInto* and *Obsolesces*. The *Document* class has an object property *hasTerm* that makes possible to associate a wiki page to each term in the grid (the co-domain is in fact the *Term* class) and, in turn, the *Term* class has an object property *hasConcept* to connect a term instance with the abstract concept it represents (the co-domain is in fact the *Concept* class). Relations among terms are expressed through the SKOS ontology while relations among concepts (and terms related to those concepts) are expressed through the tetrad. As a result, an entangled graph of relations among terms can be created by researchers, with their different competencies and interests.

The AcumeWiki engine processes this ontology to automatically generate both the wiki interfaces and infoboxes connected to such properties, following the rules described in the previous sections. Fig. 5 shows a sample wiki page about a term.


Term:Language Semantics-Pragmatics (Computer Science)

Contents [hide]
1 General Definition
2 Language
3 Semantics
3.1 Natural Language
3.2 Machine languages

General Definition

The terms semantics and pragmatics are often used in Computer Science for disciplines that examine and discuss the Web. In particular are referred to the Semantic Web, that is an evolving extension of the World Wide Web in which web content can be expressed not only in natural language, but also in a format that can be read and used by software agents, thus permitting them to find, share and integrate information more easily. The

Language semantics-pragmatics (Computer Science)		
McLuhanian Tetrad		
enhances simulation of reality		**reverses** universal language
	Language semantics-pragmatics (Computer Science)	
retrieves lack of ambiguity		**obsolesces** interpretation

Skos Relations	
alternate terms	meaning - behavior of language
broader terms	semantics
narrower terms	programming language semantics/pragmatics

Fig. 5. Example of a page representing a term in the ACUME2 epistemological grid.

The page is composed by a textual description and an infobox summarizing the ontological properties of the term. Relations with other terms are encoded as typed links to pages representing those terms. Note that the infobox is composed by three parts that respectively show the tetrad (in the top), the SKOS relations (in the middle) and some metadata about the page itself. In fact, we also added a basic set of Dublin Core (http://www.dublincore.org/documents/1998/09/dces) properties to provide basic documental metadata over the definitions.

When editing the page, the main wiki editing textarea allows for the generation of the actual textual definition for each concept, and a number of labelled form elements provide a way to connect the term to others, both already existing and yet to be defined (as known, the wiki model can be easily used to create links to *potential* pages, i.e., pages that do not exist yet but of which we can foresee the nature and title already). Once the page is saved, the terms specified in the form elements become links to actual or yet-undefined terms that can be either explored or further defined in a never-ending expansion of the semantic space of the defined terms.

VIII. CONCLUSIONS

The strength of the OWiki approach relies on its connection with ontologies and its capability of providing users with simple but powerful interfaces. OWiki makes it possible, on the one hand, to create consistent ontological data and, on the other, to edit these data through intuitive forms and templates. The development of the tool is not complete yet. The current implementation is an integrated prototype that allows users to load and manipulate ontological data regardless of their application domain. A wider end-users evaluation of OWiki is the first item in our agenda. We plan to perform a quantitative and qualitative analysis of users' involvement in using the tool and to compare OWiki with other ontology-based solutions.

More functionalities will also be added in future releases of the system: a sophisticated interface for customizing forms, a larger set of widgets described in the GUI interface and integrable in the OWiki pages, and a more flexible importer. We also plan to study solutions for letting users to use either plain wiki textareas or templates to edit the same semantic data.

ACKNOWLEDGMENTS

The authors would like to thank Vita Fortunati, Claus Huitfeld, Dino Buzzetti and Ewa Sikora, involved in the Acume2 project. The authors also recognize endless credit to Silvia Duca and Valentina Bolognini for their previous works about Gaffe.

REFERENCES

[1] Berners-Lee T, Hendler J, Lassila O (2001) The Semantic Web. In Scientific American, pp. 34-43

[2] Munat B (2004) The Lowercase Semantic Web: Using Semantics on the Existing World Wide Web

[3] Buffa M, Gandon F, Ereteo G et al (2008) SweetWiki: A semantic wiki. In Journal of Web Semantics, v. 6(1), pp. 84-97

[4] Völkel M, Krötzsch M, Vrandecic D et al (2006). Semantic wikipedia. In Proceedings of the 15th international conference on World Wide Web (WWW 2006), pp. 585-594. Edinburgh, Scotland

[5] Koren Y (2008) Semantic forms. http://www.mediawiki.org/wiki/Extension:Semantic_Forms

[6] Schaffert S (2006) IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'06), pp. 388-393. Manchester, UK

[7] Passant A, Laublet P (2008) Towards an Interlinked Semantic Wiki Farm. In 3rd Semantic Wiki Workshop (SemWiki2008), co-located with ESWC2008. Tenerife, Spain

[8] Bolognini V, Di Iorio A, Duca S et al (2009) Exploiting Ontologies To Deploy User-Friendly and Customized Metadata Editors. In Proceedings of the IADIS Internet/WWW 2009 conference. Rome, Italy

[9] McLuhan, M. (1988). Laws of Media: The New Science. University of Toronto Press