A Semantic Wiki Based Light-Weight Web Application Model

Jie Bao[1], Li Ding[1], Rui Huang[1], Paul R. Smart[2], Dave Braines[3], and Gareth Jones[3]
[1]Tetherless World Constellation, Rensselaer Polytechnic Institute, Troy, NY, USA
[2]School of Electronics and Computer Science, University of Southampton, Southampton, UK
[3]Emerging Technology Services, IBM United Kingdom Ltd, Winchester, Hampshire, UK
{baojie,dingl,huangr3}@cs.rpi.edu, ps02v@ecs.soton.ac.uk,
{dave_braines,garethj}@uk.ibm.com

Abstract. Wikis are a well-known Web 2.0 content management platform. The recent introduction of semantic wikis extends the capabilities of conventional wikis by allowing users to edit and query structured semantic annotations (e.g., categories and typed links) rather than plain wiki text. This new feature, as shown in this paper, supports the provision of a novel, transparent, and light-weight social Web application model. This model enables developers to collectively build Web applications using semantic wikis, supporting such capabilities as data modeling, data management, data processing and data presentation. The source scripts and data of such applications are transparent to Web users. Beyond a generic description for the Web application model, we show two proof-of-concept prototypes, namely RPI Map and CNL (Controlled Natural Language) Wiki, both of which are based on Semantic MediaWiki (SMW).

1 Introduction

The success of social Web applications (often called "Web 2.0" applications), such as Twitter, Wikipedia and Facebook, is evidenced by the ever-increasing mass of dynamic Web content, contributed by millions of users. Unlike conventional Web applications, where the content is primarily static and is exclusively contributed by the websites' owners, the content of social Web applications is largely dynamic and results from the collaborative contributions of multiple users. These successful social Web applications share at least two common features:

• *Simple publishing*: a user can create, edit and publish a Web page without knowing much about Web technologies, such as HTML and Web servers. For example, a Web form or a What-You-See-Is-What-You-Get (WYSIWYG) editor hides the details of HTML Web page editing; an "upload" button typically hides the details of uploading and publishing Web pages to a Web server.

• *Social interaction*: content publishing can be the result of user participation: users can collaboratively compose and improve one article on Wikipedia, or they can update their status and opinions with their friends on Facebook. Such an interactive social content contribution mechanism promotes a network effect where the value of a service provided by a user increases as more people benefit from the service [5].

Among successful Web 2.0 platforms, wikis, as exemplified by Wikipedia, are among the best known mechanisms for supporting collaborative content creation on the Web. "The Wiki Way" [11] emphasizes the principle that the content of a wiki page should

be collaboratively written using some *simple* markup languages in Web browsers, and user collaboration is expected to grow and improve the content.

Social Web applications, including wikis, usually offer limited support for user contributed structured content. For example, a blog is usually submitted via a Web form with a fixed set of properties like title, content and tags. Although users may assert tags to annotate the semantics of a Web 2.0 page, they cannot declaratively publish the detailed structure or semantics of the content. Moreover, users often have to follow a fixed user interface design to access the structured annotations (e.g. author and date) of published content. For example, the posts in Craigs' list (www.craigslist.org) apparently contains latent structures (e.g., "2004 honda civic 2dr,..."), but users can only use text search to locate or filter posts of interest. This limitation leaves Web users with limited means for preserving the structure of data to (i) avoid unnecessary overheads in natural language understanding and (ii) leverage smart services (such as semantic search and inference) that utilize the preserved semantics.

A number of efforts have been recently observed in addressing the above limitations with Semantic Web technologies. In particular, *semantic wiki* systems, such as Semantic Mediawiki (SMW) [9] and IkeWiki [13], have been developed to extend conventional wikis by providing support for simple semantic annotations on wiki pages, such as categories and typed links. A wiki page in semantic wikis may contain both conventional wiki text and structured data, and the structured data can be further accessed by customizable queries using simple query languages.

By supporting both *annotation* and *querying* of structured data on wiki pages, semantic wikis may serve as a platform for light-weight data modeling, computation and presentation tasks that are traditionally out of end users' control. Semantic wikis, therefore, offer the promise of a new application model with a couple of interesting characteristics:

• *Rich data modeling*: User contributed content may be a mixture of text and structured data, and semantic wikis can best preserve structured data without forcing structured representation of the free text part. With the structured data, a semantic wikis can function like a light-weight database or a knowledge base, and users can model data using several common modeling methods, e.g., relational modeling or rule modeling.

• *Transparent data processing*: as wikis allow simple computing logics (such as declaring an object and applying a data processing rule) to be published as a part of wiki pages in the forms of wiki scripts, they are transparent to all wiki users and can be collaboratively authored and improved in Web browsers.

• *Social programming*: The transparency of data modeling and data processing and the convergence model of wiki themselves opens up the development of Web applications to all interested users.

In this paper, we provide a generic description of a semantic wiki based Web application model. We then present two proof-of-concept prototypes, namely RPI Map and CNL (Controlled Natural Language) Wiki, both of which are based on SMW. The main contributions of the paper are the following:

– Identification of a light-weight Web application development model that possesses the aforementioned characteristics (Section 2);

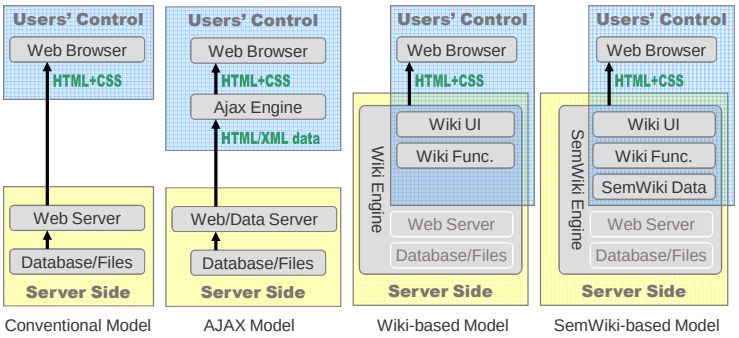

Fig. 1. A comparison of several Web application models

– Working prototypes that embody the identified model using the SMW platform (Section 3 and 4). In particular, we show that templates in SMW are useful in supporting some common data modeling and data processing tasks.

Although our description and demonstrations are limited to SMW-based implementations, the identified model is not necessarily limited to SMW or wiki-based implementations. We note that our demonstrations still carry some limitations which are mainly related to the SMW implementation. As more and more Web 2.0 applications are provided with semantic extensions (e.g., Drupal[1]), we believe the conclusions reached in this paper will also extend to other platforms.

2 Semantic Wiki Based Web Application Model

In this section, we introduce a semantic wiki based Web application model in the context of the evolution of Web application models.

2.1 Comparison of Web Application Models

The advance of Web technology drives the evolution of Web application models. Starting from just being able to browse Web pages, Web users are now able to control content publishing with the help of Web 2.0 technologies. Wikis, blog systems (e.g., Drupal and Wordpress) and similar online content management systems further provide extensible computing infrastructures that support scripting and/or customizable plugins to facilitate collective Web application development. Recent advances in the social semantic Web, such as SMW, allows users to collaboratively control structured data management. In Figure 1, we compare several Web application models.

In the *Conventional Model*, a Web application is composed of three clearly-separated major components, namely the Web browser, the Web server and the backend storage system (e.g., a database or a file system). Users of such an application are provided with limited control over the contents of the application; interaction is usually limited

[1] http://drupal.org/project/rdf

to browsing and search. The representation, computation and presentation components are primarily hosted on the server and are controlled by webmasters only.

Other models have extended the Conventional Model with extra client-side control over data and computation. The *AJAX Model* [3], which uses an AJAX engine to act as a mediator between the browser and the server, is becoming increasingly popular due to its powerful client side computing ability. It improves the user experience with respect to both data transfer (e.g., asynchronous data retrieval from the server without interfering with page display) and data presentation. For example, a powerful word processing system (e.g., Google Docs) can be used within a browser where the data is actually stored on the Web. It is notable that users may also insert client side scripts into Web applications for customized processing.

The *Wiki-based Model* enables end users to directly control some data content and presentation on the server side. For example, Wikipedia articles are collaboratively maintained by users and complex wiki templates are frequently used to enable advanced page layout (e.g., to render a calendar). A user may also call the extensions of a wiki platform (e.g., "parser functions" in MediaWiki, the system used by Wikipedia) to perform certain computation tasks such as string processing, mathematical computation and visualization. It is also notable that a wiki page may embed external script languages (e.g., JavaScript) for more advanced tasks.

Both the AJAX model and the Wiki-based model increase the user's control over data processing. The *SemWiki* (Semantic Wiki)*-based model* further users additional control on the management and consumption of structured data. For example, in Wikipedia it is not yet possible to assert a structured, queryable annotation for individual wiki page, ; neither is it possible to execute a query that "all European countries that have female government leaders". Semantic wikis address these limitations by extending wikis with the ability to create and query structured annotations using a relatively simple modelling and querying language. As a result, users can now exert greater levels of control over the data in an application. In particular, the SemWiki model supports a comprehensive in-browser scripting environment, such that a light-weight Web applications can be built collectively with high transparency on computational logics (as computational scripts are included in wiki pages). By "light-weight" we mean that the data structure, data processing logic and user interface of the application are relatively simple. In what follows, we elaborate on the components, as well as several design patterns, of the SemWiki based model.

2.2 Data Modeling

Semantic wikis are often built upon RDF triple stores for storing structured data. Thus, data in a semantic wiki does not need to be stored with a pre-defined schema (although it is possible to do this) as is required by an RDBMS. This conforms to the open nature of the Web and enables significant flexibility and extensibility in data modeling. Note that the semantic wiki model allows hybrid modeling with predefined "schema", schema-free user added metadata, and unstructured data, thus making the extension of an application much easier. For example, users can always add new attributes as needed to a specific article in addition to existing attributes.

Some semantic wikis (like SMW) not only preserve the semantic structure of data, but also provide light-weight query capabilities (with a role similar to that of SELECT queries in SQL). For example, in SMW it is possible to pose a query

```
{{#show [[Category:Article]][[tag::<q>Category:food</q>]] }}
```

to find all articles tagged with "food" or its subtags (like "donut").

Note that since the modeling specification and queries themselves are also presented as semantic wiki pages, they can be accessed, updated or deleted in the same fashion as other wiki pages in the browser. Thus, semantic wikis function as a virtual abstraction layer over the Web server and database/file systems, such that programmers are not required to directly access the layers hidden below semantic wikis. This characteristic naturally enables the collective construction of an application.

2.3 Data Processing

Several MediaWiki extensions provide scripting functionalities similar to that of the basic constructs of a programming language. When combined with templates, semantic annotations and semantic queries, these extensions can be used to support a wide range of light-weight data processing abilities. Some of the most useful extensions include[2]:

• **Variables**: General variables are supported by the Variable Extension so that users can name a long expression as a variable, and then reuse it later on the same wiki page. A special type of wiki pages called "template" pages also allow the use of variables as input parameters.

• **Datatypes**: The String Functions extension provides some common string functions such as string length and concatenation; the Array Extension provides array operations (e.g., search and sort) and set operations (e.g., union and intersect) on arrays.

• **Control Flow**: The Parser Functions Extension offers: (i) expression evaluation, e.g., evaluation of mathematical expression like "(1+2)", and logical expressions like "(true and false)"; and (ii) conditional statements such as IF-THEN-ELSE conditions. The Loop Extension supports loop structures such as WHILE and DO-WHILE.

2.4 User Interface

In SMW, many elements of a user interface (UI) in an application can be constructed using scripts. For example, the Semantic Forms[3] extension offers a form-based editing interface for users to edit template-based data. Utilizing templates and queries also allow users to control the look-and-feel of the user interface and present the data with various visual elements (e.g., table, picture and tree). Since templates and forms are also wiki pages and can be edited in browsers, the design and subsequent improvement of the UI are also supported by collective scripting enabled by the semantic wiki model.

In addition, in MediaWiki (thus, also in SMW) users can also inject JavaScript code into a wiki page, either by including server-side scripts or code in some client-editable special wiki pages. Some SMW-based applications (e.g., wikicafe.metacafe.com and metavid.org) have developed advanced UIs such as video browsing and annotation.

[2] See http://tw.rpi.edu/wiki/ASWC2009Bao#Links for their URLs

[3] http://www.mediawiki.org/wiki/Extension:Semantic_Forms

By aggregating the data management and data processing features with JavaScript, designers are able to design highly interactive, visual interfaces for the manipulation of semantically enriched data.

2.5 Strength and Limitations

By supporting data modeling, processing and presentation (via a user interface) abilities, semantic wikis provide a transparent platform for light-weight Web application development. In particular, such a development model enjoys several advantages:

• **Flexibility**: Because contents and scripts are both stored as wiki pages, users can always read and update them directly through browsers. Thus, the improvement to both contents and the application (as constructed with scripts) becomes a dynamic, highly portable, and easily accessible process.

• **Socialization**: Semantic wikis inherit the collaborative nature of wikis, in particular the support of social user participation, e.g., user login, collaborative editing, and revision history. This may encourage large-scale, collaborative interactions between users.

• **Inference Ability**: The availability of semantically enriched content in semantic wikis makes it possible to do some inference with data, thus supporting potentially improved means of data consumption (e.g., search and query).

Nevertheless, it should be noted that the semantic wiki based model also has some limitations:

• **Efficiency**: Semantic wikis often use a triple store for data storage. The state-of-the-art in triple stores has not yet reached the same level of maturity and scalability as that of relational databases. This may present some efficiency problems for applications with very large numbers of wiki pages. In addition, the overhead of parsing and rendering structured data in semantic wiki pages often results in delays in response. Performance tuning for commercial deployment is thus often crucial.

• **Modeling Ability**: The native modeling support of semantic wikis is usually limited to a subset of RDF or OWL. The page-centric structure of semantic wikis also makes the modeling of complex knowledge and data structures difficult. Thus, building an application that requires very complex data structure or logic with the semantic wiki based model can be challenging. This will be further discussed in Section 5.

• **Safety**: As wikis in general are designed to be an open collaborative environment, support for safety control is usually minimal. For Web applications requiring stronger access control to avoid malicious changes to the content of the application, additional efforts are required to ensure data and application safety[4].

It should be noted that these limitations are mainly related to the current implementation of semantic wiki systems (like SMW), not to the general semantic wiki based model we have presented. We believe that many of these limitations will be overcome or alleviated with future advances in the state-of-the-art.

In the next two sections, we will introduce two concrete examples of Web applications based on SMW, namely RPI Map and CNL Wiki. These applications illustrate

[4] We noticed some recent advances in SMW access control, e.g., the HaloACL extension from Ontoprise.

how SMW enables light-weight data modeling, data processing and user interface development with an open, extensible architecture.

3 Case Study: RPI Map

This section introduces **RPI Map** (http://map.rpi.edu), a SMW-based Web application that exemplifies the general methodology we described in the previous section.

RPI Map is a campus map application for the Rensselaer Polytechnic Institute (RPI) community. It integrates and visualizes location based information, such as buildings, events and classes, on an interactive map using the Google Map API[5]. At its core is a Semantic MediaWiki installation along with a set of mediators that perform data mashup from multiple external data sources. In what follows, we will describe in detail how SMW supports the implementation of RPI Map.

3.1 Data Modeling

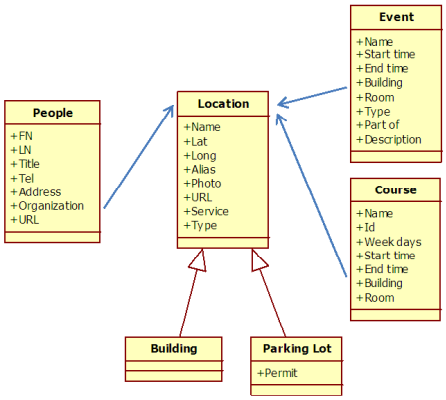

Fig. 2. Data Schema of RPI Map

Template as Schema. In RPI Map, templates play an important role in data organization; they serve as a "virtual" schema for relevant data. The main types of data on RPI Map include locations (and its subtypes such as buildings and parking lots), people, events, courses, campus shuttle routes and real time shuttle positions. Much of this data is published by various individual entities across RPI. For example, event information is published as an RSS feed of the institutional calendar, course information is available from the RPI catalog as a text table, and people information is provided as downloadable vCard files from the RPI directory. To integrate these data in RPI Map, for each external data source there is a mediator (implemented using server-side scripts[6]) to transform the original data into a form that can be consumed by the wiki platform. These sources are then linked by various semantic queries based on location information (e.g., name of a specific building).

Templates are used as the general output format of these mediators. Each template corresponds to a type of data in the system and describes a set of attributes that relevant data instance must possess. For example, `Template:LocationInfo` defines a

[5] http://code.google.com/apis/maps/

[6] It is also possible to use client-side script to do data importing, thus users may add other types of data to the system.

template with location-relevant parameters, e.g., name, latitude, longitude and aliases. Together, these templates define a "schema" to organize data in the application. This schema is shown in Fig. 2.

Note that while RPI Map uses schema-like templates for data modeling, these templates should not be understood as a relational schema in the database domain. These templates provide, on the basis of the triple-based data representation infrastructure of SMW, a higher level abstraction of some related triples. It is not a requirement in RPI Map to have all data fit in a rigidly defined schema, neither need an instance of a template be associated with only data that are required by the template. Such an ability brings additional flexibility in accommodating data from heterogenous data sources.

3.2 Data Computation

Stored Query. Many queries are repeatedly used in many different components of RPI Map. For example, one commonly used query is used to map a location based on its aliases. Such a query is stored as a template page `Template:Alias`:

```
{{#ask: [[Has alias::{{PAGENAME}}]] |link=none|limit=1}}
```

It may be embedded in other pages that need such a query.

Thus, templates can play a role similar to that of stored procedures in a relational database. As those templates can be edited in the browser by users (with some necessary protection mechanisms), they are more transparent and easier to access than stored procedures (which are normally hidden behind a server-side DBMS).

Data Cleansing. SMW can also help clean up corrupt or inaccurate data in the course of integrating data into RPI Map. For example, in transforming people information (in the vCard format), the same location (e.g., a person's office address) may be referred to by different names by different university departments. In addition, new variations of a location's name may be discovered when new data is added (e.g., from the event RSS feed). A special name recognition template was designed, partly leveraged by a fuzzy string similarity comparison parser function, to identify the closest known location or its aliases.

3.3 User Interface

Query-based Map Generation. Each of the map pages on RPI Map is based on some semantic queries. For example, the "Today Event" page relies on a query in the form[7]:

```
{{#vardefine: eventLocations|
   {{#ask:[[Category:Event]]
     [[has end time::> {{LOCALMONTHNAME}}
       {{LOCALDAY}},{{LOCALYEAR}} 00:00]]
     |?has location=|mainlabel=-|link=none}} }}
{{#map_objects:
  {{#ask: [[Has alias::
     {{#var:eventLocations}}]]
```

[7] For ease of presentation, the query is simplified from the actual query.

```
| ?Has LatLong
...
|limit=200|link=none }} }}
```

where `map_objects` is a function that will automatically generate a map via the Google Map API based on the result of the "ask" semantic query. The query asks for "all the locations (potentially in their alias forms) of today's events and their latitude/longitude (some other attributes omitted)". Note that the example also demonstrates the use of variables in constructing complex queries.

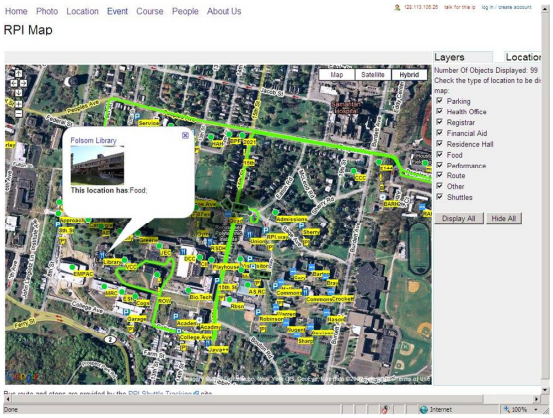

Fig. 3. RPI Map Main Interface

Integration with JavaScript. In RPI Map, JavaScript is used together with wiki scripts. Some examples are:

- Popping up a new window with additional information about a location, e.g., its full name, picture and services;
- Generating labeled markers and customized icons;
- Validating user-input location information;
- Displaying geographic information in an extant data format, e.g., Keyhole Markup Language (KML).

The main page of RPI Map is shown in Fig. 3.

Part of RPI Map source code for data representation and navigation has been released as the Tetherless Map Extension to Mediawiki[8].

4 Case Study: CNL Wiki

CNL Wiki (http://tw.rpi.edu/proj/cnl) is another application that conforms to the architecture described in Section 2. The CNL Wiki aims to provide a user friendly interface for collaborative ontology building. The wiki application is built on top of SMW, thereby inheriting its collaborative nature, high portability and accessability. In addition, a Controlled Natural Language (CNL) is used to provide some support for ontology development in OWL. The aim is to improve the comprehensibility of generated knowledge statements for end users. In this section, we will introduce in details how SMW enables the representation of strongly structured data (i.e., OWL knowledge bases), data computation (e.g., CNL sentence generation), and user interface generation. Additional details about CNL Wiki can be found in [1].

[8] http://www.mediawiki.org/wiki/Extension:Tetherless_Map

4.1 Data Modeling

In this subsection, we introduce how SMW templates can be used in modeling structured data and generating semantic data.

Modeling Structured Data. In order to accommodate ontology construction in OWL within SMW, we need to address a number of expressivity constraints associated with SMW. Currently, SMW does not provide full native support for OWL modeling. In order to address this limitation, we developed a meta-model extension to SMW, called SMW-mOWL (where "m" stands for meta model). Please refer to [1] for a complete description of SMW-mOWL.

SMW-mOWL represents an OWL ontology using a set of wiki pages, each of which encodes some ontology elements (i.e., classes, properties, individuals and axioms) as wiki template instances. For example, suppose we have an OWL statement saying that "every father is a person that has a child who is also a person", which can be given in the OWL Abstract Syntax (OWL-AS) as:

```
Class(Father partial Person restriction
    (hasChild someValuesFrom(Person)))
```

This statement can be broken down into several template instances and represented as SMW pages. For example, on the page "Category:Father", the above statement in OWL-AS is represented with three template instances:

```
{{NamedClass          |label=Father          |plural=Fathers  }}
{{NamesClassRelation |type=subClassOf       |class=Person  }}
{{someValuesFrom      |on property=hasChild |on class=Person  }}
```

Thus, each category page represents a single class in OWL along with some axioms about the class. The "`Template:NamedClass`" describes annotations to this class, such as comments and natural language labels. "`Template:NamedClassRelation`" describes relationships between two classes (here a class inclusion relationship). "`Template:someValuesFrom`" represents a restriction that the class in question must satisfy.

Semantic Data Generation. The use of a template-based mechanism for SMW-mOWL also allows us to store the knowledge model in the SMW database (tuple store). For example, an instance of `Template:someValuesFrom` will be persisted as an instance of the ternary property `owl:someValuesFrom` in the wiki of which the first element is the class where the template instance resides, the second element is the "on property" parameter, and the third element is the "on class" parameter. Such persisted data in the database can be further consumed by other scripts, e.g. for CNL generation (see the next subsection), or external tools, e.g., a SPARQL query engine.

4.2 Data Computation

Once the SMW-mOWL meta-model is persisted in the database, the query language for SMW (SMW-QL) can be used to retrieve specific information from the model, which can be consumed by other wiki scripts. In this subsection, we describe two such usage patterns.

Templates as Functions. To query information stored in the SMW tuple store, we use a set of templates to implement queries and perform some additional processing. Templates, here, are used in a role similar to that of functions in a conventional programming language. For example, the `Template:CNL.Rabbit.getLabel` takes as input the name of a page (denoted as {{page}}), and performs the following queries:

- query if the page is an anonymous class using

```
{{#ask: [[:{{page}}]][[Category:Anon]]
   |format=list|limit=1|link=none }}
```

  The query result will be stored as a variable (it is empty (false) iff the class is an anonymous class).
- If it is an anonymous class, call `Template:CNL.Rabbit.Anon` to construct its label in the Rabbit CNL, otherwise return its label by calling a SMW query:

```
{{#ask: [[:{{page}}]]  |?CnlLabel= |mainlabel=-
                       |format=list|limit=1|link=none }}
```

CNL Generation. Queried results from the SMW database will be further parsed and processed by a set of CNL generation templates. Currently, we support two CNLs in English, namely Rabbit [4] and Attempto Controlled English (ACE) [7].

For example, the `Template:CNL.Rabbit.getSomeRestrictionAssertion` template generates Rabbit CNL sentences based on the "someValuesFrom" restrictions of a class "{{page}}". It performs the following tasks:

• Use `Template:CNL.Rabbit.getLabel` to get the natural language label of the class in question (i.e., the input parameter "{{page}}").

• Use a query to fetch all "someValuesFrom" restrictions related to "{{page}}":

```
{{ask: [[{{page}}]]   |?owl:someValuesFrom
  |mainlabel=-|format=list|link=none}}
```

• For each such restriction, parse its "on property" and "on class" values, then use `Template:CNL.Rabbit.getLabel` to get their natural language labels and generate Rabbit sentence using the Rabbit grammar. For instance, for the example in the last subsection, we will have "Every Father has child Person.".

A meta-model template like `Template:NamedClass` may call a CNL generation template, such as `Template: CNL.Ace.Concept` (which will, in turn, call other templates to construct all CNL sentences about a specific class). Thus, users will get CNL description of a knowledge statement whenever the statement is created by users using form-based editing techniques, or whenever an external ontology is imported into the wiki. Fig. 4 shows a property in an ontology using the Rabbit CNL.

Fig. 4. A property represented in the Rabbit CNL

4.3 User Interface

Structured data representation in SMW also supports user interface construction. This is again facilitated by (semantic) templates.

Controlling Page Layout. Similar to conventional wikis, templates in semantic wikis play important roles in controlling page layout. For example, `Template:Property` controls the look-and-feel of a property, such as

- Content organization (e.g., as tables), color schema, font size and other visual elements of a page;
- Linking to the editing interface;
- CNL statements in selected CNL languages, each in a separate table section.

In contrast to conventional templates, a template in SMW is able to use semantic queries so that content from other pages can also be displayed on the page in question. In addition, by separating the text content and semantic content of a page, SMW is able to partially reuse a page's content, and it does not need to keep the original layout of the content of other pages. These features are not available by conventional page inclusion in MediaWiki.

Light-weight GUI. Using semantic queries, structured information across multiple pages can be aggregated on one page using graphical representations. One example on the CNL Wiki is a query-based class hierarchy tree. The template `Template:GUI.Tree` defines a recursive query that fetches class inclusion relations from a root class in a specified ontology. For example, the following script creates a hierarchical treeview presentation of all subclasses of the Animal class in the "Rabbit Ontology".

```
{{GUI.Tree |root=Category:Animal |ontology=Rabbit Ontology  }}
```

We believe this approach can be extended to support the display of other types of GUI element, e.g. toolbars, menus and listboxes.

Form Generation. By utilizing the "Semantic Forms" extension of SMW, some template instances can be edited using a form-based interface. The generation of such forms can be automated from the template definition. Thus, having the template-based OWL meta-model immediately provides us with a light-weight OWL ontology editor within the SMW environment. Each form comprises some controls (textboxes, checkboxes, radio buttons, and so on) that support various editing operations. Auto-completion (which may in turn involves some queries) in semantic forms allows sentence editing using existing entities in the ontology.

5 Discussion and Related Work

5.1 Collaborative Web Application Development

A recent review [6] on the trends of Web application development analyzed several popular Web development models and showed how they benefit from technological advances and the evolution of user behavior. The review showed that collective intelligence can benefit not only content creation but also application development. Our semantic wiki based model clearly exemplifies this trend via its emphasis on general-purpose, in-browser scripting that enables users to contribute to the representational,

computational and visualization capabilities of the target system. Thus, an application could be extended in a collaborative fashion as the result of the activities of multiple individuals, e.g., by adding new information sources via the creation of client-side mediators, creating new datatypes and associated templates, and making data available to other systems by the creation of new export formats.

A number of semantic wikis (e.g., AceWiki [10] and IkeWiki [13]) and Semantic Web platforms (e.g., HyperDE [12, 14] and social semantic desktop [2]) have been used to support Web application development following a similar approach to that adopted by ourselves in the proposed application model. These efforts share common characteristics in that they all allow social publishing of semantically enriched data. Our approach, in contrast to these efforts, enables users to contribute both semantically-enriched data, as well as data consumption scripts. Both these capabilities are supported using a simplified, easy-to-use, browser-based publishing process, to collectively build Web applications.

5.2 Users Participation

The semantic wiki based model we propose has some advantages compared to the extant Web collaborative programming approaches, including off-browser approaches like Concurrent Versions System (CVS), and in-browser approaches like Bespin[9]. The built-in support for Semantic Web features makes our approach easier to build knowledge-intensive applications. In addition, since the components of an application (e.g. data structure and UI) can be all presented as wiki pages, they can be edited via the usual wiki editing interface without requiring a special client devolvement software. This may encourage the social participation of users in improving the application, therefore better exploiting the network effect.

We may observe in the proposed model, as in Wikipedia, a "long tail" effect [8], such that while a large proportion of edits are performed by a small, core group of "elite" users, the aggregation of small numbers of edits from the majority of "common" users also contributes a significant portion of contributions. It has be shown that although Wikipedia was driven by the influence of elite users early on, there has been a subsequent shift towards contributions from the common users [8]. We expect the same social participation pattern to occur in the case of semantic wiki based application models: there may be an elite group that is initially involved in designing the templates, data models and UI of the application, while the majority of users interact mainly with forms-based entry and prebuilt queries, but also occasionally contribute to the improvement of the application. Yet, it is notable that this is different from the traditional programming paradigm where developers and users are two distinctive groups. In the semantic wiki based model, the boundary of the two groups is not absolute and the change of roles is easy.

5.3 Data Modeling

The data modeling ability in semantic wikis is a tradeoff between the complexity and flexibility of data models. Semantic wikis extend traditional wikis with the ability to add

[9] http://labs.mozilla.com/projects/bespin/

and query metadata. However, the core extension to conventional wiki scripts is quite small and thus is relatively easy to learn. Compared with the conventional techniques for developing Semantic Web applications, the semantic wiki approach is limited in providing the ability to model heavy-weight semantic structure, e.g., the page-centric representation of wiki modeling makes it sometimes hard to freely create knowledge statements. On the other hand, this simplicity also makes the semantic wiki model easier to learn and to use; it therefore lowers the threshold for mass user participation.

It should be noted that while the demonstrated examples in this paper only use form-based data entry from users, the semantic wiki based model does not exclude other forms of editing, e.g., by using an ontology browser in the Halo extension[10] or a video stream editor of the MetaVidWiki extension[11].

When compared with the conventional Web model where data is organized with a pre-defined database schema, the semantic wiki based model shows a clear advantage in terms of its support for the RDF graph model. This is evident in applications where database schemas tend to be too rigid and too slow to evolve, as both requirements, user expectations and data structures are often consistently changing. In semantic wiki based modeling, relationships between data elements can be represented explicitly and transparently as an RDF graph, thus making it easier to create, extend and combine data, and for an application to utilize unanticipated data sources (e.g., to add a new GeoRSS source to RPI Map), and vice versa[12].

6 Conclusions

In this paper, we present a light weight Web application model based on semantic wikis. The model utilizes the data modeling, processing and presentation abilities of semantic wikis, which enable better flexibility, social interaction and inference ability in building a Web application compared with conventional models. We illustrate our approach with two proof-of-concept applications, RPI Map and CNL Wiki, both of which are based on Semantic MediaWiki (SMW). Using the two examples, we show that semantic queries and templates are useful building components in realizing many of the data modeling, processing and presentation abilities of semantic wikis.

Our future work will focus on the enhancement of the aforementioned prototype systems. The extensible architecture of the two applications allows them to evolve with user contributed scripts. For example, in CNL Wiki, we plan to add additional CNL verbalization support by new sets of CNL templates, and an ontology repository management ability by using a set of ontology templates. The ultimate goal is to better demonstrate how to create and update an application using semantic wiki thereby encouraging the adoption of the proposed semantic wiki based development model.

Acknowledgement This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement

[10] http://semanticweb.org/wiki/Halo_Extension

[11] http://metavid.org/wiki/MetaVidWiki

[12] This paragraph is influence by a presentation “Drupal and the Semantic Web” by Jamie Taylor on Jun 6th, 2009. http://bit.ly/2H5Pil

Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

We thank Jin Guang Zheng for part of the RPI Map implementation and Zhenning Shangguan for part of the CNL Wiki implementation.

References

1. J. Bao, P. R. Smart, D. Braines, G. Jones, and N. R. Shadbolt. A controlled natural language interface for semantic media wiki. In *2nd Annual Conference of the International Technology Alliance, http://tw.rpi.edu/wiki/TW-2009-05*, 2009.
2. S. Decker and M. R. Frank. The networked semantic desktop. In C. Bussler, S. Decker, D. Schwabe, and O. Pastor, editors, *WWW Workshop on Application Design, Development and Implementation Issues in the Semantic Web*, volume 105 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
3. J. J. Garrett. Ajax: A new approach to web applications. adaptivepath.com, February 2005. [Online; Stand 18.03.2008].
4. G. Hart, M. Johnson, and C. Dolbear. Rabbit: Developing a control natural language for authoring ontologies. In *ESWC*, pages 348–360, 2008.
5. J. A. Hendler and J. Golbeck. Metcalfe's law, web 2.0, and the semantic web. *J. Web Sem.*, 6(1):14–20, 2008.
6. M. Jazayeri. Some trends in web application development. In *FOSE*, pages 199–213, 2007.
7. K. Kaljurand and N. E. Fuchs. Bidirectional mapping between owl dl and attempto controlled english. In *PPSWR*, pages 179–189, 2006.
8. A. Kittur, E. Chi, B. A. Pendleton, B. Suh, and T. Mytkowicz. Power of the few vs. wisdom of the crowd: Wikipedia and the rise of the bourgeoisie. In *Presentation at 25th Annual ACM Conference on Human Factors in Computing Systems (CHI 2007)*, 2007.
9. M. Krötzsch, D. Vrandecic, M. Völkel, H. Haller, and R. Studer. Semantic wikipedia. *J. Web Sem.*, 5(4):251–261, 2007.
10. T. Kuhn. Acewiki: A natural and expressive semantic wiki. *Semantic Web User Interaction Workshop at CHI 2008*, 2008.
11. B. Leuf and W. Cunningham. *The Wiki way: quick collaboration on the Web*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.
12. D. A. Nunes and D. Schwabe. Rapid prototyping of web applications combining domain specific languages and model driven design. In *ICWE*, pages 153–160, 2006.
13. S. Schaffert. Ikewiki: A semantic wiki for collaborative knowledge management. In *WETICE*, pages 388–396, 2006.
14. D. Schwabe and M. R. da Silva. Unifying semantic wikis and semantic web applications. In *International Semantic Web Conference (Posters & Demos)*, 2008.