Abstract: To become a part of forthcoming Semantic Web, it will be necessary to add semantic technologies into existing web applications. In our effort to extend existing web applications for the Semantic Web, the aim of our research is to address the critical and challenging problem of preserving role separation in development and maintenance of such applications. We present our solution and demonstrate its use.

Keywords: semantic web, migrating, extending, JSP, RDFa, role separation, MVC, annotation, maintenance

1. Introduction

Semantic Web gradually transforms from myth to reality. Some predictions say that by the end of year 2012, 80% of public Web sites will use some level of semantic hypertext to create semantic Web documents. [13] Soon it will be necessary to enable existing Web applications for the emerging Semantic Web in order to avoid being left behind. That means that all those applications that already constitute World Wide Web (WWW) should be enriched with semantic technologies. Many of those applications are commercial applications. During its lifetime it is expected that commercial applications provide revenue for the owners. A number of published sources have reported that software maintenance costs typically constitute more than half the total cost spent on a software system during its useable life. [17] Usually, when software maintenance costs exceed its income, it is the end of product lifetime and all future development actions are stopped. [16] Therefore, the goal is to minimize maintenance costs to prolong product lifetime. Empirical studies show a close relationship between software maintenance and activities undertaken during software development, for example the influence of the selected system development methodology. [8][6] The maintenance effort is considered to be acceptable for those applications whose architecture persists module separation and design follows principles of object-oriented programming. Recommended architecture for Java web applications is Model-View-Controller (MVC) due to its ability to be easily extended and maintained [7].

Recently the World Wide Web Consortium (W3C) has proposed Resource Description Framework attributes (RDFa) – a syntax that allows embedding Resource Description Framework (RDF) information into (X)HTML documents. RDFa could have an important role in the grand plan for the Semantic Web, in which web page data is readable not only by humans, but by automated processes as well. Intense research is being done in extending frameworks and approaches for semantic web development. [2][15][18] Results of that effort are precondition to more significant research in field of web applications migration to semantic web.

In this paper we present a method and tool for integrating RDF into existing web applications. We have tried to find a method that would require minimal changes to the existing code, and that would take full advantage of MVC architecture.

This document is structured as follows: in chapter 2 we highlight general principles of Java web application architecture and design which should be followed and preserved during application adaptation process for the semantic web. Semantic web technologies used in our approach are briefly introduced in chapter 3. In chapter 4 we describe importance of having vocabulary expert included in semantic web application development process. Possible embedment strategy of RDFa in existent application is also shown. Chapter 5 is a review of existing annotation approaches and appeared problems. In chapter 6 we present our solution of previously identified problems. Finally, in chapter 7 and 8 we summarize our work and give directions for future work.
2. Web Application architecture

The focus of this paper is on Java EE (Java Enterprise Edition) applications based on MVC Model 2 pattern. It is the template for web applications suggested in almost every book on Java Web technologies, so it is not surprising that it has become so popular over the years and is rooted in the minds of developers. In MVC architecture, a controller handles system navigation, a model stores a set of data, and a view presents the data stored in the model. Java web applications implement an MVC pattern using a servlet for the controller, a Java Bean (or EJB) as its model, and JSP pages as the view. This pattern has many benefits including role separation, single point-of-entry and high scalability. Some very popular Java web frameworks are based on MVC pattern, such as Struts, Tapestry, JSF, Spring MVC etc., so we can with certainty say that MVC pattern is widely spread around.

Role separation is an important feature of MVC pattern. Today it is not important only that applications function well, but also that they have an appealing interface. Usually, designing web pages is not programmers’ forte, and on the other hand, most web designers have little or none programming skills. So the answer is to separate the layers of content generation (programmer) and content presentation (page designer). MVC architecture provides just that. Some other role separation is also possible, for example the data-access layer could be entrusted to a developer who is more familiar with database (optimizing queries).

For purpose of this paper we will concentrate on separation between presentation and domain model layers. To achieve our task, annotating JSP pages with RDFa tags, we are proposing a solution that includes vocabulary (ontology) expert in the development without deteriorating role separation.

3. Introduction to RDFa

W3C proposed a concept for semantic description of web information – Resource Description Framework (RDF). [9][10] Resources (e.g. web pages, books, and people) are described by values of specified properties (e.g. language, number of pages, names). Information is a statement – a triple composed of a subject (resource), a predicate (property), and an object (value or another resource).

If we represent subjects and objects as nodes, and predicates as links, the whole document can be represented by a graph.

Embedding RDF in XHTML is enabled using an extension of XHTML – RDFa. RDFa syntax provides mapping to RDF triples and thus enables triples to be extracted from well formed RDFa document. [11][12]

RDFa attributes (existing XHTML attributes and RDFa-specific attributes) are listed below:

- **about** – a URI specifying the resource the metadata is about
- **rel** and **rev** – specifying a relationship or reverse-relationship with another resource
- **href**, **src** and **resource** – specifying the partner resource
- **property** – specifying a property for the content of an element

Optional attributes:

- **content** – attribute that overrides the content of the element when using the property attribute
- **datatype** – attribute that specifies the datatype of the content
- **instanceof** – attribute that specifies the RDF type(s) of the subject (the resource that the metadata is about).

The only attribute for setting a subject is @about. The attributes for setting a predicate are @rel, @rev and @property, while the attributes for setting an object are @resource, @href, @content, and @src. Attribute @instanceof sets both a predicate and object at the same time.

Many namespaces (vocabularies) can be used to describe information, like those to describe calendar events, those to describe people, bibliographic resources etc.

In a field of our interest is FOAF (Friend of a Friend) – namespace for describing people [4]. FOAF defines classes such as Person, Document, Image, alongside some properties of those things such as name, mbox (mailbox), homepage, birthday etc., as well as relationships between FOAF members (for example depiction, that relates e.g. Person to a Image).

vCard vocabulary (represents electronic business card) is also used throughout our solution in order to verify usage of multiple vocabularies in RDFa.
4. RDFa tagging

4.1. Introducing vocabulary expert - annotator

During the relatively short history of computers and software, the deployment profile for an application, as well as the computer and software technology and most important the number of technologies used to develop such an application has changed. If an application developer wants to build an average Java web application on her own, she should be familiar with at least these technologies (languages): (X)HTML, CSS, JavaScript, Java, JSP, SQL or one of ORM (object relational mapping) tools and in nowadays Ajax.

In order to build semantic enriched application, developer should learn new semantic web technologies. Even though it seems that all you need to build RDFa aware web application is knowledge of RDFa alone, that is not true. To be able to use and understand it, one should also know RDF and be familiar with existing vocabularies (eg. FOAF, vCard, etc…).

RDFa is syntax for expressing structured data in XHTML. Hence, to annotate web application data, changes of presentation layer have to be made by adding RDFa tags. Considering role separation, one developer does not have to be an expert in all of the above mentioned technologies. Page designer is an expert in CSS, HTML etc., and she builds a clean interface. Who should be assigned to incorporate dynamic content into interface depends on the application design and framework used. Usually it is a Java expert who built the domain model.

We believe that data semantic annotation is extensive job and should be left to vocabulary expert (annotator). Correct annotation requires knowledge of main semantic web technologies including available vocabularies and their meanings.

4.2. Incorporating RDFa tags

There is a difference between web site and web application. The key difference is that web site has a purpose to deliver information; and it is very important that the information is understandable to the wide audience. Considering that semantic web is a vision of web where information has well defined meaning, it is not surprising that most recommendations and studies on using semantic web technologies are mainly based on web sites and they are not completely applicable for web applications.

Main purpose of web application is to perform some functions based on user inputs; hence web applications are designed with much more user interaction. Based on the user input, data is generated and resulting page is served to the user. Pages have very little static data (written in XHTML), majority is dynamically generated. As a result, there is no appropriate solution to embed RDFa tags in XHTML.

```html
<tr>
  <td>% student.getCode() %</td>
  <td>
    <span property="foaf:firstName">
      % student.getFirstName() %
    </span>
  </td>
  <td>% student.getLastName() %</td>
</tr>
```

**Figure 1 RDFa – defining predicate**

Embedding predicate statement into dynamically generated XHTML is as simple as adding a font color to given data. (See Figure 1)

What complicates writing RDFa tags is defining statement subject (See Figure 2 RDFa – defining subject).

To be able to add right content to subject it is necessary to know data exactly. In our example one should know precisely which student is described to be able to fill the subject (@about).

The only feasible solution is to generate tags during the generation of related data. Obviously, some Java code is going to be needed to generate RDFa tags.

```html
<tr>
  <span about="#Lidia">
    <td>% student.getCode() %</td>
    <td>
      <span property="foaf:firstName">
        % student.getFirstName() %
      </span>
    </td>
    <td>% student.getLastName() %</td>
  </span>
</tr>
```

**Figure 2 RDFa – defining subject**
5. Annotation approaches

5.1. Semantic document annotation

RDFa is a new W3C recommendation, so approaches in that precise field are still in research phase. However, semantic document annotation is being researched for some time now and that research has resulted in several annotation tools and approaches.[5][3][14] Those researches have helped in identifying main web site annotation problems. Here, we will reflect just on those that also affect annotation of web application data.

Main identified problems are:

1. Manual annotation is expensive – only valuable information should be annotated; it is necessary to help human annotator with this task.
2. Object identification - general problem stems from the fact that without corresponding tool support, annotators would too often create new objects rather than reuse existing ones.
3. Vocabulary (ontology) changes – in real world ontology changes over time so there is a need for continuous annotation adaptation of new semantic terminology. [14]

5.2. Annotation using RDFa

RDFa on rails¹ is a solution that helps users in RDFa tags adding process to applications developed using Ruby on Rails. As far as we know, it is the only annotation solution that uses RDFa.

To add RDFa tags using this library you only have to change the view of your application, whilst controller and model are left untouched. A shortcoming of this approach is manual addition of method calls to your view. Depending on number of pages it could be time consuming and error prone task. It could happen that same data on different views is mapped to distinct properties and lead to inconsistency.

6. Implementation

The embedded business logic and requirements in the code of the application discourage application owners from rewriting it to make use of semantic web technologies. Such rewrite would require a great amount of time and effort. [1] There is a need for solution that would make minimal changes to existent code and would fulfil this task.

In order to ease future maintenance we have set requirements that our solution has to accomplish. Requirements are set according to the arguments stated in introduction and shortcomings of previous annotation approaches:

- preserve role separation
- preserve layer separation
- provide automatic tag generation
- enable easy vocabulary embedding
- vocabulary adaptability

We have designed a mapping tool SmartMap. Mapping is done between domain object model and vocabularies. Application dynamic content (contained in presentation layer) is provided by object method calls (eg. student.getFirstName()). For that reason we have decided to use approach in which we map methods to properties.

Figure 3 shows the screenshot of SmartMap tool. Left pane holds all public classes and methods of chosen domain model, while right pane holds classes and properties of needed vocabularies. Mapping is achieved by clicking corresponding elements from left and right pane. Object identifiers are needed to be set once per class. Besides defining relationships between properties and methods, for each used object (class) identifier has to be given. Object identifier will constitute subject of RDF statement. Using object identifiers assures that object will be reused across application.

Additional information about method-property mapping, like specified tag to wrap mapping, data type, content and language are also provided. RDFa provides possibility to define language of annotated data. Assuming that application uses some kind of internationalization, you could specify method which returns used language to set it to RDFa tag.

One of the advantages of our solution is the feature that wherever method is used, it will be exactly annotated. It promises completely consistent annotations.

It is expected that schema descriptions (vocabularies) change over time to reflect changes in the world. To carry out according changes in application extended using our approach there is no need to make any changes.

¹ http://rdfs.rubyforge.org/
in the code, just to regenerate classes using SmartMap tool. In our approach vocabulary expert does not need to have any additional knowledge of technologies used throughout development. If domain object model follows Java naming convention whole mapping process could be done by vocabulary expert alone. In reality programmer familiar with domain model should be consulted to explain the meaning of certain methods.

6.1. Implementing generated code

After the mapping process is finished, XML code presenting mappings, and classes needed to enrich presentation layer with defined semantic are generated. For each mapped class appropriate wrapper class is generated using decorator design pattern. Java programmer uses those classes together with some API classes we provide to integrate our solution (Figure 4 and Figure 5).

Generated class StudentSemWeb extends class Student. Methods mapped to vocabulary properties are overridden and they return annotated data.

Solution provided by tool such as SmartMap ease application migration towards semantic
web. Majority of code needed to add semantics to application is generated and can be regenerated in case of some vocabulary change, which simplifies further application maintenance.

7. Future work

Although SmartMap is presently standalone application it should be realized as a plug-in for Java development tool. The advantage is that all changes performed by the user through the process of migration and maintenance will be immediately synchronized between domain object model and objects generated by the tool.

RDFa is suitable primarily for web sites, so we have encountered several problems in our effort to integrate RDFa tags throughout our application.

One of the most common problems is to define semantic dependence between data dynamically generated on the client side.

Other problems arise due to still present vagueness and immaturity of RDFa syntax.

Resolving these challenges is going to be the basis of our future work.

8. Conclusion

In this paper, we presented possible solution of integrating RDFa tags into existing Java application based on MVC design pattern. We identified main requirements that have to be satisfied in order to assure easy maintenance of application extended with semantic web technology. We presented our mapping tool that is realized in accordance to mentioned requirements. During development and usage of our tool, we revealed some challenges and limitations of RDFa syntax; and we have provided directions for future work.

9. References