A Framework-based Approach for the Integration of Web-based Information Systems on the Semantic Web

Danillo R. Celino
Ontology & Conceptual Modeling Research Group (Nemo) - Department of Informatics, Federal University of Espírito Santo (Ufes)
Vitória, ES, Brazil
drcelino@inf.ufes.br

Luana Vettler Reis
Ontology & Conceptual Modeling Research Group (Nemo) - Department of Informatics, Federal University of Espírito Santo (Ufes)
Vitória, ES, Brazil
luanna.vettler@gmail.com

Beatriz Franco Martins
Ontology & Conceptual Modeling Research Group (Nemo) - Department of Informatics, Federal University of Espírito Santo (Ufes)
Vitória, ES, Brazil
bfmartins@inf.ufes.br

Vítor E. Silva Souza
Ontology & Conceptual Modeling Research Group (Nemo) - Department of Informatics, Federal University of Espírito Santo (Ufes)
Vitória, ES, Brazil
vitor.souza@ufes.br

ABSTRACT

For the vision of the Semantic Web to become a reality and its benefits harnessed, data available on the Web must also be published in the form of *linked data*. Moreover, the quality of the abstract conceptual models behind this data, i.e., their ontology, can also have a big influence in the adoption of linked data sets and their vocabularies. In this paper, we propose *FrameWeb-LD*, an approach for the integration of Web-based Information Systems on the Semantic Web, which uses well-founded languages and methods for the modeling of ontologies and aids developers in publishing their application's data and services on the *Web of Data*.

Keywords

Web Engineering; Frameworks; Semantic Web; Linked Data; Ontologies.

1. INTRODUCTION

The Semantic Web vision, first described by Berners-Lee et al. [3] in their seminal article, proposed to harness the architecture of the World Wide Web to link data instead of just documents, adding semantics (meaning) to such links. According to the authors, making data available on the Web in a machine-processable format would allow the creation of software agents that could aid us in tasks that are repetitive, impractical or even impossible to accomplish nowadays.

Such tasks are hindered by the amount of data available on the Web. From product specifications to geographical information, from scientific research results to governmental data, an increasing number of people and organizations are choosing to share their data with others, contributing to a *data deluge*. This phenomenon creates problems such as how to provide access to data so it can be most easily reused; how to enable discovery of relevant data within the multitude of data sets; or how to integrate data from different and formerly unknown data sources [12]. The first step is to publish all this data in the form of *linked data* [2].

According to [13], however, the WWW is not the only source and inspiration for the technologies that support the Semantic Web. The construction of abstract models (Conceptual Modeling, Ontology Engineering) and computing with knowledge (Logic and Artificial Intelligence) are also involved in the process of building for this *Web of Data*.

Ontologies have a fundamental role on the development of the Semantic Web [15]. They define a common meaning to data published in various data sources, helping with their reutilization, discovery and integration. The more disseminated these vocabularies, the more likely they are to be used. We believe a key factor in making these ontologies popular is their quality which, in turn, is tightly connected to the quality of the languages, methods and tools used in their definition.

This paper proposes *FrameWeb-LD*, an extension of the *FrameWeb* [19, 22] method that aids developers in making their Web-based Information Systems (WIS) — both data- and service-wise — available on the Semantic Web, i.e., published as linked data. The main problem we aim to address is that of adoption: by providing a systematic method based on well-founded ontologies, coupled with tools that automate certain parts of the process, we facilitate the task of integrating a WIS into the *Web of Data*, with higher quality models, thus promoting the adoption of *linked data*. Of course, this is a small contribution regarding a broader problem of realizing the Semantic Web vision. We can, however, harness the benefits of *linked data* even if such vision has not been (or will never be) reached.

The remainder of the paper is divided as following: Section 2 summarizes the baseline of our work; Section 3 presents our proposal, *FrameWeb-LD*; Section 4 describes our proposal's evaluation; Section 5 discusses related work; and, finally, Section 6 concludes.

WebMedia '16, November 08 - 11, 2016, Teresina, PI, Brazil

ISBN 978-1-4503-4512-5/16/11...$15.00

DOI: http://dx.doi.org/10.1145/2976796.2976847

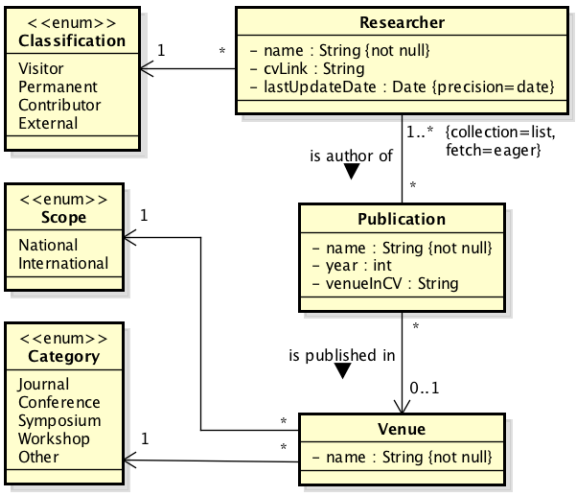

Figure 1: Example of *FrameWeb* Domain Model.

2. BASELINE

FrameWeb-LD is an extension of *FrameWeb* [19, 22], a *Framework-based Design Method for Web Engineering. FrameWeb* is motivated by the advantages brought by the use of frameworks and container-based architectures (e.g., Java™ Enterprise Edition [4]), such as avoiding the continual rediscovery and reinvention of basic architectural patterns and components, reducing cost and improving the quality of software by using proven architectures and designs [21].

FrameWeb incorporates concepts from well established types of framework — e.g., Front Controller (MVC), Object/Relational Mapping (ORM), Dependency Injection (DI) — into architectural design models. For instance, Figure 1 shows part of a domain model for C2D, a Web-based Information System (WIS) that keeps track of members of a post-graduate program of an university and their respective publications, used as a running example in this paper. In addition to the usual UML constructs (visibility kinds, cardinalities, data types, etc.), the diagram shows ORM mappings such as *not null*, *precision=date*, *fetch=eager*, etc. [22].

There are three other models prescribed by *FrameWeb*: the *Persistence Model* specifies the persistence operations that should be implemented using the ORM framework for each domain class; the *Navigation Model* shows the different elements that compose an MVC solution for the presentation layer (web pages, forms, MVC controllers, etc.); finally, the *Application Model* models the service layer and determines how the DI framework should connect the controllers to service classes that, in turn, depend on the Data Access Objects (DAOs [1]) from the persistence layer.

In its original proposal [22], *FrameWeb* already includes a Semantic Web extension called *S-FrameWeb*, which proposes the inclusion of a domain analysis [7] activity for the construction of an ontology [9] of the domain, the use of the Ontology Definition Metamodel (ODM)[1] for domain models and an extension of the MVC framework to deliver results in OWL[2] upon request. Our proposal, *FrameWeb-LD*, intends to replace *S-FrameWeb* for the reasons explained next.

S-FrameWeb does not prescribe a systematic method for the construction of the ontology. In this work, we propose the use of SABiO, a Systematic Approach for Building Ontologies [6], which organizes the ontology construction process in five phases: (1) purpose identification and requirements elicitation; (2) ontology capture and formalization; (3) design; (4) implementation; and (5) test. These phases are supported by well-known activities in the Requirements Engineering lifecycle, such as knowledge acquisition, reuse, documentation and evaluation.

Moreover, ODM defines a language that focuses on operational ontologies (as defined in [6]) in OWL. Following SABiO, we propose the use of the well-founded language OntoUML [10] for ontology capture and formalization. Tools such as OLED[3] and Menthor[4] can aid modelers in designing OntoUML diagrams, plus include features for deriving OWL operational ontologies from OntoUML models.

Finally, *S-FrameWeb* included linked data (LD) support for a single MVC framework based on technology that is now outdated. We propose: (1) the use of tools such as D2RQ[5], which serve as an LD adapter layer over relational databases (predominant database type in WIS); (2) to follow best practices in LD publishing [12]; and (3) to aid developers in providing Semantic Web Services [20] using standard description languages such as OWL-S[6]. Tools such as OWL-S Editor [5] and OWLComposer[7] can help in this matter.

The above Semantic Web technologies are based on the RDF[8] (Resource Description Framework) data model, which describes resources on the Web using triples, i.e., statements composed by three parts, subject-predicate-object, forming node-and-arc-labeled directed graphs [12]. Triples can be stored in a special kind of database called *triplestore* and be queried via the semantic query language SPARQL.

3. PROPOSAL

In this section we present our proposal of a new semantic extension for *FrameWeb*, called *FrameWeb-LD*. The contributions of this work are: (a) an extension of the metamodel of *FrameWeb* allowing linked data mappings to be represented in its design models; (b) the integration of the systematic approach SABiO for building ontologies with the ontologically well-founded language OntoUML; (c) a tool for the automatic generation of code, relieving developers of most of the effort in publishing the aforementioned linked data and semantic web services.

The flowchart in Figure 2 provides an overview of the development process proposed by *FrameWeb-LD*. Dashed lines represent the flow of information, whereas solid ones denote the sequence of tasks (besides the usual flow of information between two sequential activities). The process is divided in five stages, indicated by the diagram's swim-lanes (names on the left-hand side of the figure). It is important to note that while the flowchart indicates a sequence of activities/phases, we do not prescribe a specific development life-cycle. We suggest, however, the use of iterative and agile processes.

The phases of the *FrameWeb-LD* development process are detailed in the subsections that follow. We use the generic

[1]http://www.omg.org/spec/ODM/1.1/.
[2]http://www.w3.org/standards/techs/owl.
[3]http://nemo.inf.ufes.br/projects/oled/
[4]http://www.menthor.net/
[5]http://d2rq.org/.
[6]http://www.w3.org/Submission/OWL-S/.
[7]http://sourceforge.net/projects/owl-scomposer/.
[8]https://www.w3.org/RDF/

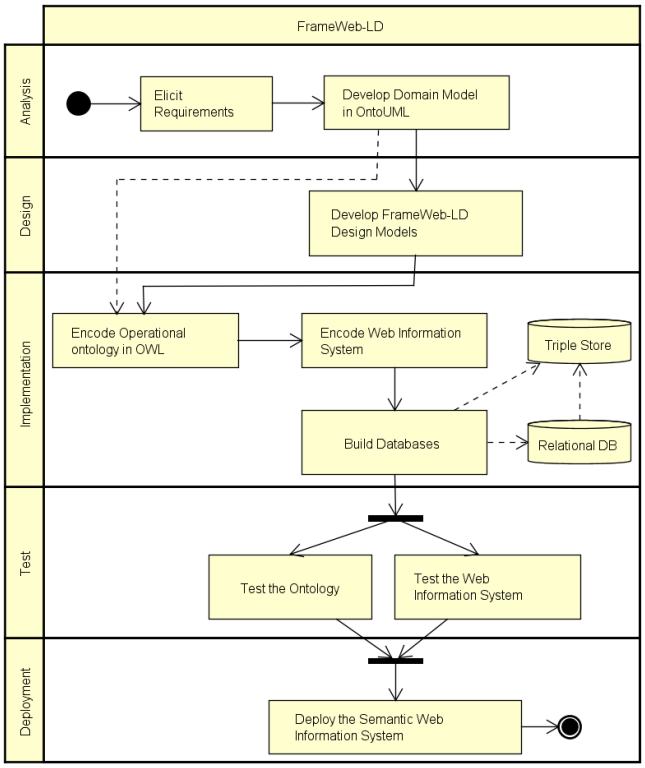

Figure 2: Overview of the *FrameWeb-LD* process.

role *developer* to represent the actors conducting the activities in different stages of the software process (which could be requirements engineers, software architects, programmers, etc., depending on the stage).

3.1 Analysis

During this phase, the developer conducts the first two steps of the SABiO process: (1) purpose identification and requirements elicitation (*Elicit Requirements* activity); and (2) ontology capture and formalization (*Develop Domain Model in OntoUML* activity). The output of this phase is an ontology that represents the concepts of the problem domain of the WIS, modeled in OntoUML.

As an extension of *FrameWeb*, these activities should integrate with the usual Requirements Engineering (RE) process conducted to develop the WIS. As part of it, requirements engineers usually produce a conceptual model of the problem at hand. With *FrameWeb-LD*, the OntoUML model produced at this stage can be used as basis for the construction of the conceptual model during RE. Optionally, the developer could conduct a Domain Analysis activity [7], broadening the ontology scope to the entire domain of the problem (e.g., publications from researchers associated to post-graduate programs) and not just the problem at hand (metrics used at a specific program for evaluating researchers).

For our running example, we focused on the problem at hand. In parallel with ordinary RE activities (e.g., capture of functional and non-functional requirements), we elicited Competency Questions [8] focusing on our post-graduate program — such as "What is a researcher in the post-graduate program?" (CQ1), "What are the possible roles for a researcher?" (CQ2), or "What is the scoring system to evaluate researchers in the program?" (CQ8) — and produced an ontology in OntoUML, shown in Figure 3.

This conceptual model shows elements from the domain, such as the post-graduate program, its researchers, their publications and how they are scored following a scoring system. Each class has a stereotype that determines their relation to the concepts of the foundational ontology UFO. For a complete description of OntoUML and what each of these stereotypes mean, the reader should refer to [10].

Later, this model was used as basis for a UML class diagram (with added elements such as attributes, enumerations, etc.), featured in the software requirements specification document for C2D.

3.2 Design

At the design phase, the developer should produce the *FrameWeb* models described in Section 2, namely: *Application*, *Domain*, *Navigation* and *Persistence* models. The *FrameWeb-LD* extension proposes additions to the first two on this list, described next.

3.2.1 Domain Model

The *Domain Model* is based on the ontology/conceptual model built in the previous phase, but with added details regarding implementation. As shown in Figure 1, *FrameWeb* adds Object/Relational Mapping (ORM) annotation to domain classes, besides other usual implementation details (attribute data types, navigability of associations, etc.). Analogously, *FrameWeb-LD* adds linked data (RDF) mapping annotations to this model. The resulting model serves also as Ontology Design Specification, which is the expected result of the design phase of SABiO.

The meta-models that define the *FrameWeb* language [19] were, thus, extended to allow the inclusion of RDF annotations, which specify how the data from the WIS relates to well-known vocabularies from the Semantic Web, with the purpose of integrating them into the Web of Data [12]. Due to space constraints, a fragment of these meta-model extensions is shown in Figure 4. The complete meta-model is available at the *FrameWeb* project website.[9]

White classes in the figure come from the UML meta-model extended by *FrameWeb* [19], which includes the `FrameWebModel` class, in blue, and `DomainAttribute`, green. The remaining classes, in yellow, are the extensions from the *FrameWeb-LD* meta-model, which are based on the OWL 2.0 syntax specification.[10] The `VocabularyModel` class represents the *FrameWeb-LD* model that contains linked data annotations. Such model can import `Axiom`s and `Annotation`s from external `Vocabulary` given their URI (the meta-model uses `IRI`, following the term from the OWL 2.0 syntax specification, although we prefer the more popular term, URI). Then, the model is annotated via `VocabularyAssociation`, `VocabularyProperty` or `VocabularyConstraint`.

The proposed extensions are illustrated in a partial *Domain Model* for C2D in Figure 5 as follows:

- Although not shown in the diagram, the meta-model associates vocabulary identifiers (IDs) to their respective URIs, just as the header of an RDF document does. In the example, `foaf` is associated with http://xmlns.com/foaf/0.1/ (Friend of a Friend vocabulary)

[9] http://nemo.inf.ufes.br/projects/frameweb/
[10] https://www.w3.org/TR/owl2-syntax/

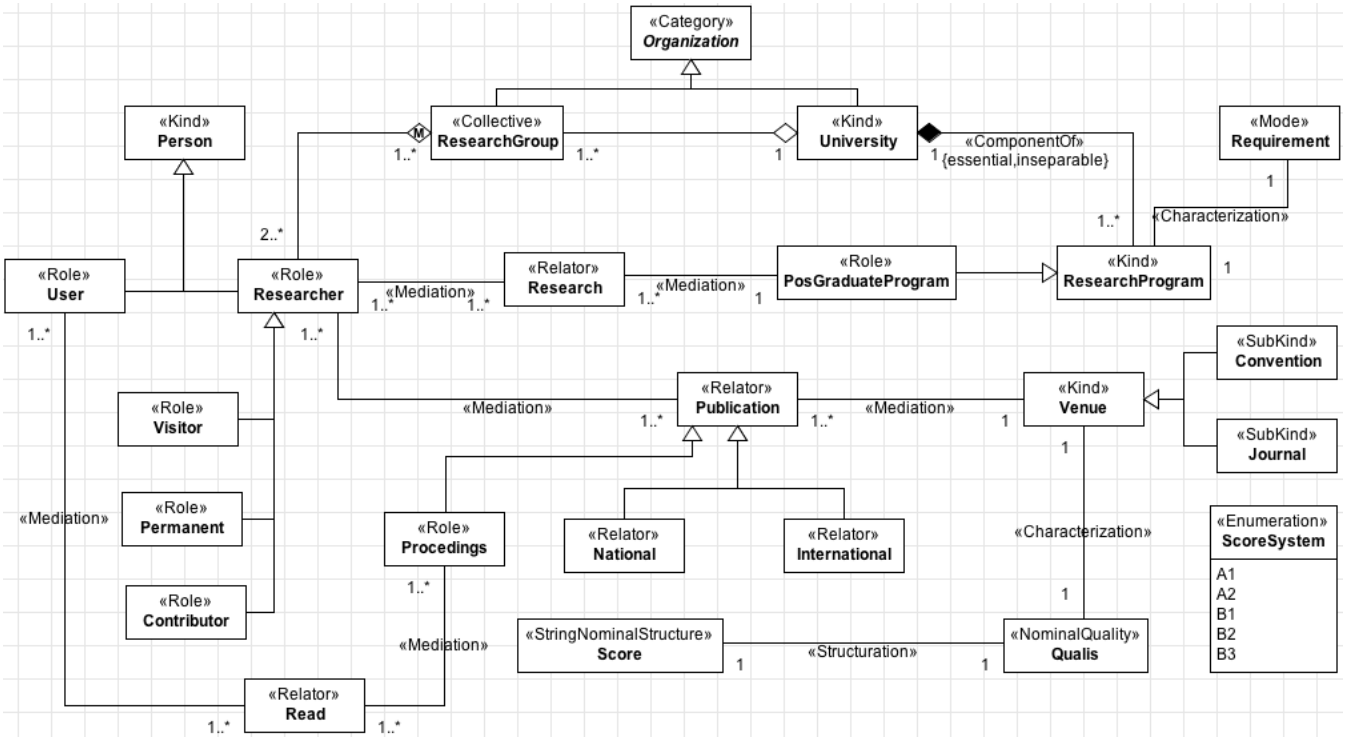

Figure 3: OntoUML ontology for our running example, C2D.

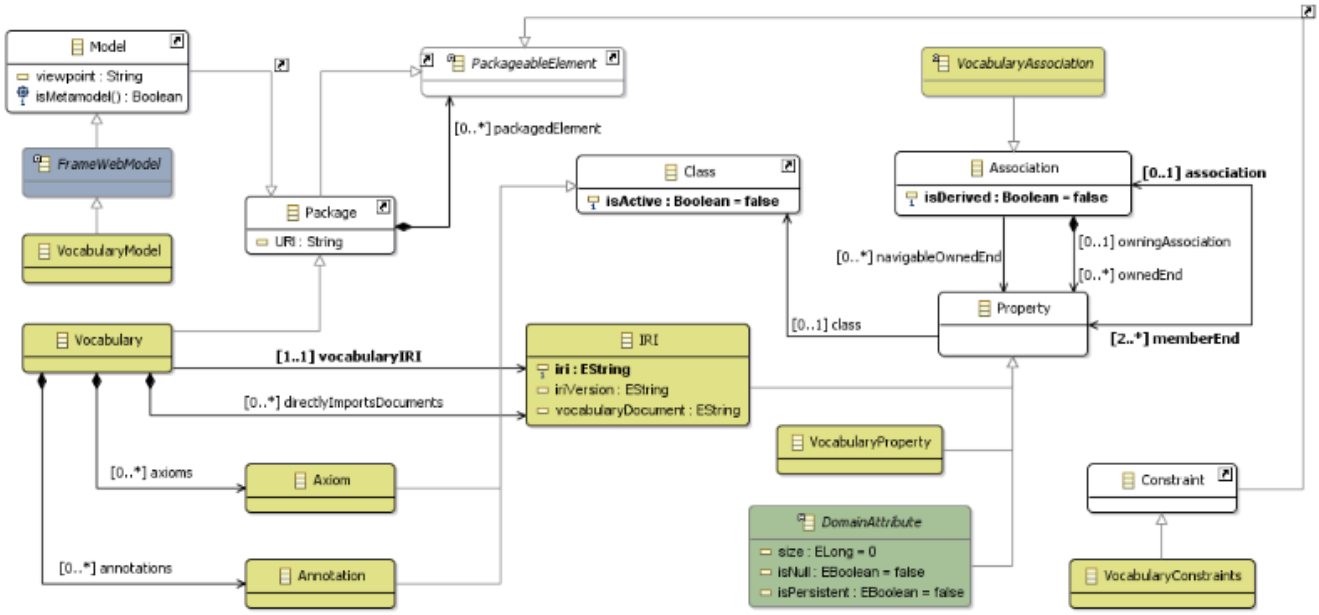

Figure 4: Fragment of the *FrameWeb-LD* meta-model.

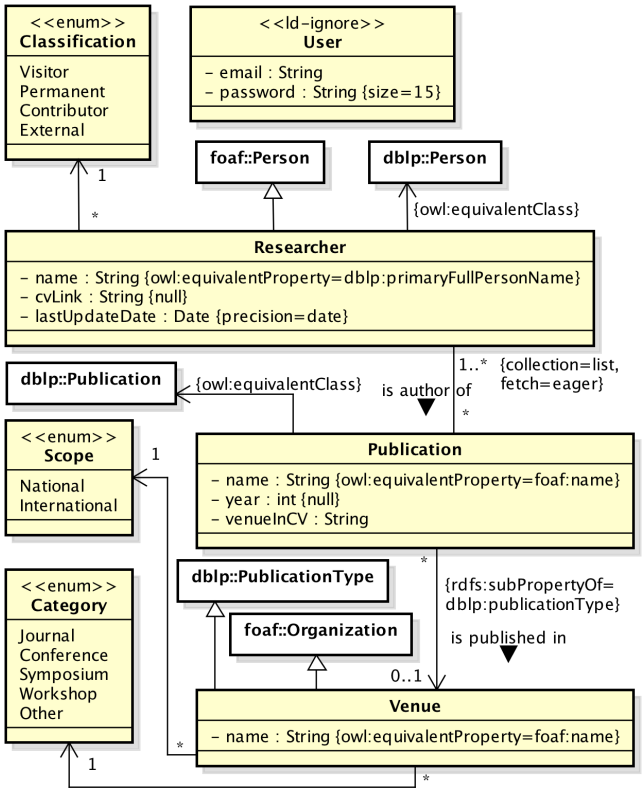

Figure 5: A *FrameWeb-LD* Domain Model.

and `dblp` with http://dblp.rkbexplorer.com/id/ (DBLP Computer Science Bibliography dataset);

- Classes from external vocabularies are shown using their vocabulary IDs as UML namespace (denoted with two colons `::` instead of just one `:` as in RDF). They can be related to classes from the WIS via UML associations, navigable towards the external class, representing an RDF triple: the class from the WIS is the subject, the external one is the object and the predicate is specified as a constraint. In the example, `Researcher` is `owl:equivalentClass` to `dblp:Person`;

- As a syntactic sugar, the `rdfs:subClassOf` relation between a class from the WIS and one from an external vocabulary can be represented by a UML inheritance association. In the example, `Researcher` is `rdfs:subClassOf foaf:Person`;

- Triples concerning attributes of classes are represented using constraints in the form *predicate=object*. In the example, `Researcher.name` is `owl:equivalentProperty` to `dblp:primaryFullPersonName`;

- Constraints in associations between classes from our WIS establish relations among object properties (in the same way constraints in attributes establish relations among data type properties). In the example, the association between `Publication` and `Venue` is `rdfs:subPropertyOf dblp:publicationType`;

- Last, but not least, data from all classes are to be published as linked data, unless the `ld-ignore` stereotype is used (either to exclude specific attributes or entire classes). In the example, the `User` class is excluded from the linked data set to be published.

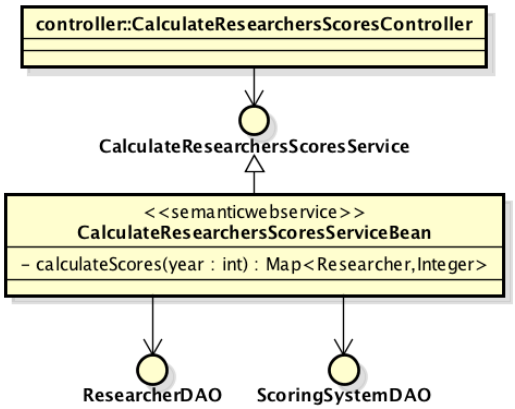

Figure 6: A *FrameWeb-LD* Application Model.

In the same spirit as *FrameWeb*, *FrameWeb-LD Domain Models* give clear instructions on how to publish linked data from our WIS in the next phases of the process.

3.2.2 Application Model

The *Application Model* is centered on classes that implement the services of the WIS, which are made available to (human) users via a Web interface mediated by the MVC framework. For software agents, however, a more suitable way to access these services are Semantic Web Services [20].

Therefore, *FrameWeb-LD* proposes the `semanticwebservice` stereotype to be used in application classes as a whole or just some of their methods. Figure 6 shows part of a model from C2D that specifies that the methods from the service class responsible for calculating the scores for researchers of the post-graduate program in a given year should also be made available via Semantic Web Services.

As with the *Domain Model*, the information in this diagram is used in the next phases of the process to guide the implementation of the WIS.

3.3 Implementation

We propose three activities for this phase: *Encode Operational Ontology in OWL* (the equivalent to SABiO's *Implementation* phase), *Encode Web Information System* and *Build Databases*.

The first activity can be automated by tools such as OLED or Menthor Editor (cf. Section 2), which can generate OWL operational ontologies from OntoUML models. This generated OWL file is the base for the vocabulary (RDF schema) of our WIS, but needs to be completed with relations to external vocabularies, represented earlier as RDF annotations on the *Domain Model* (e.g., Figure 5).

To ease this task, we built a prototype of a code generator tool called *ReMaT* (**Re**lational Database **Ma**pping to **T**riple Store) that reads a *FrameWeb-LD Domain Model* and produces the triples that complete our WIS' vocabulary. An example is shown in Listing 1. In this excerpt from the OWL file generated by Menthor, *ReMaT* adds the `rdfs:subClassOf` and `rdfs:subPropertyOf` relations present in Figure 5.

Listing 1: Excerpt from operational ontology in OWL generated by Menthor and *ReMaT*

```
<owl:Class rdf:about="http://dev.nemo.inf.ufes.br/owl
    /c2d.owl#Publication">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/
      XMLSchema#string">Publication</rdfs:label>
  <rdfs:subClassOf rdf:resource="http://dblp.uni-
      trier.de/rdf/schema-2015-01-26#Publication"/>
</owl:Class>

<owl:Class rdf:about="http://dev.nemo.inf.ufes.br/owl
    /c2d.owl#Venue">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/
      XMLSchema#string">Venue</rdfs:label>
  <rdfs:subClassOf rdf:resource="http://xmlns.com/
      foaf/0.1/Organization"/>
  <rdfs:subClassOf rdf:resource="http://dblp.uni-
      trier.de/rdf/schema-2015-01-26#PublicationType"
      />
</owl:Class>

<owl:ObjectProperty rdf:about="http://dev.nemo.inf.
    ufes.br/owl/c2d.owl#isPublishedIn">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/
      XMLSchema#string">isPublishedIn</rdfs:label>
  <rdfs:domain rdf:resource="http://dev.nemo.inf.ufes
      .br/owl/c2d.owl#Publication"/>
  <rdfs:range rdf:resource="http://dev.nemo.inf.ufes.
      br/owl/c2d.owl#Venue"/>
  <rdfs:subPropertyOf rdf:resource="http://dblp.uni-
      trier.de/rdf/schema-2015-01-26#publicationType"
      />
</owl:ObjectProperty>
```

The codification of the WIS is done following the *FrameWeb* method and using the contents of its proposed models together with the chosen frameworks to build the application. *FrameWeb-LD* adds Semantic Web Services to the WIS based on *Application Models*. Once the methods from the application classes are implemented, IDEs such as Eclipse[11] can generate a WSDL[12] description for the web service, which serves as input to an OWL-S tool (cf. Section 2), which finally produces the OWL-S description of the Semantic Web Service. An example is show in Listing 2 regarding the web service illustrated earlier in Figure 6.

Listing 2: Semantic Web Service description generated by OWL-S Editor.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xml:base="http://dev.nemo.inf.ufes.br/owl-s/c2d/
      CalculateReseacherScores/_Service.owl#"
  xmlns:owl="http://jamsci.servehttp.com/owlsedit/owl
      .rdf#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
      ns#"
  xmlns:rdfs="http://jamsci.servehttp.com/owlsedit/
      rdf-schema.rdf#" xmlns:service="http://staff.um
      .edu.mt/cabe2/supervising/undergraduate/
      owlseditFYP/owls11/Service.owl#">
  <owl:Ontology rdf:about="">
    <owl:versionInfo>Version 1.0</owl:versionInfo>
    <rdfs:comment>Service Ontology to Calculate
        Reseacher Scores</rdfs:comment>
    <owl:imports rdf:resource="http://www.w3.org
        /1999/02/22-rdf-syntax-ns"/>
    <owl:imports rdf:resource="http://jamsci.
        servehttp.com/owlsedit/owl.rdf"/>
    <owl:imports rdf:resource="http://jamsci.
        servehttp.com/owlsedit/rdf-schema.rdf"/>
    <owl:imports rdf:resource="http://staff.um.edu.mt
        /cabe2/supervising/undergraduate/owlseditFYP/
        owls11/Service.owl"/>
  </owl:Ontology>
  <service:Service rdf:ID="_Service">
    <service:presents rdf:resource="http://dev.nemo.
        inf.ufes.br/owl-s/c2d/
        CalculateReseacherScores/_Profile#_Profile"/>
    <service:describedBy rdf:resource="http://dev.
        nemo.inf.ufes.br/owl-s/c2d/
        CalculateReseacherScores/_ProcessModel#
        _ProcessModel"/>
    <service:supports rdf:resource="http://dev.nemo.
        inf.ufes.br/owl-s/c2d/
        CalculateReseacherScores/_Grounding#
        _Grounding"/>
  </service:Service>
</rdf:RDF>
```

[11]http://www.eclipse.org

[12]Web Services Description Language, http://www.w3.org/TR/wsdl

However, the description produced by these tools is disconnected from the vocabulary of our WIS: the tools "make up" a vocabulary which needs to be manually replaced by the developer. Having the *ReMaT* tool (semi-)automate this task is one our plans for the near future.

Finally, the last activity of this phase regards the creation of the databases that will serve our WIS. First, a relational database is created as usual (modern ORM frameworks support the automatic creation of the database schema from the system domain classes) in order to store all the data from our WIS. Besides this database, we propose the use of a triple store that can provide software agents with linked data triples with derreferenceable URIs, plus a SPARQL endpoint for querying the data.

D2RQ (cf. Section 2) creates a layer on top of the relational database and offers the aforementioned features based on a semi-automatic conversion from the database schema to RDF. Like the OWLComposer tool mentioned earlier, D2RQ creates a "mock" vocabulary that needs to be replaced by the one generated for the WIS. Our tool, *ReMaT*, replaces the vocabulary automatically, relieving the developer of another tedious task.

Listing 3 shows part of the mapping file generated by D2RQ to map the database table `Researcher` to RDF. Again, *ReMaT* completes the file with information from *FrameWeb-LD* models, such as the `rdfs:subClassOf`, `owl:equivalentClass` and `owl:equivalentProperty` relations in the listing.

Listing 3: Excerpt from the relational-to-RDF mapping file generated by D2RQ and *ReMaT*

```
@prefix c2d: <http://dev.nemo.inf.ufes.br/owl/c2d.owl
    #>

# Table Researcher
map:Researcher a d2rq:ClassMap;
  d2rq:dataStorage map:database;
  d2rq:class c2d:Researcher;
  d2rq:classDefinitionLabel "Researcher";
  rdfs:subClassOf foaf:Person;
  owl:equivalentClass dblp:Person;
  .
map:Researcher_name a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:Researcher;
  d2rq:property vocab:Researcher_name;
  d2rq:propertyDefinitionLabel "Researcher name";
  owl:equivalentProperty dblp:primaryFullPersonName;
  d2rq:column "Researcher.name";
  .
```

3.4 Test and Deployment

The *Test* phase of *FrameWeb-LD* consists of testing both the ontology and the WIS. We do not provide any contributions in this sense. The ontology should be tested as per SABiO (validating the competency questions, verifying the operational ontology, etc.) and the WIS following appropriate testing methods from Software/Web Engineering.

Deployment is done as usual for the WIS (no different than *FrameWeb*), with the addition of D2RQ, which needs to be executed alongside the Web Server that hosts the WIS.

4. EVALUATION

We have conducted preliminary evaluation of *FrameWeb-LD* with students enrolled in a *Web Development and the Semantic Web* course from the Post-Graduate Program in Informatics of our university. The students developed small Web Information Systems (WIS) using frameworks and were asked to have their WIS publish linked data and connect to external vocabularies. At the end of the semester, each group would produce a report, documenting their WIS with *FrameWeb* and *FrameWeb-LD* models.

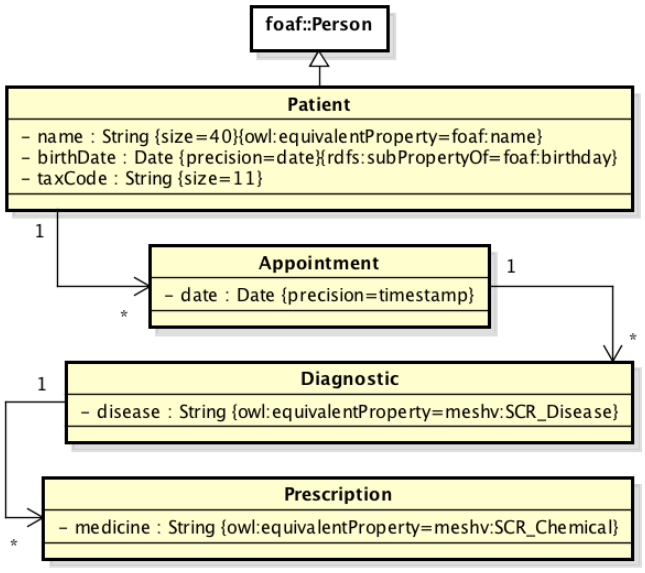

Figure 7: Domain Model for *Medic*.

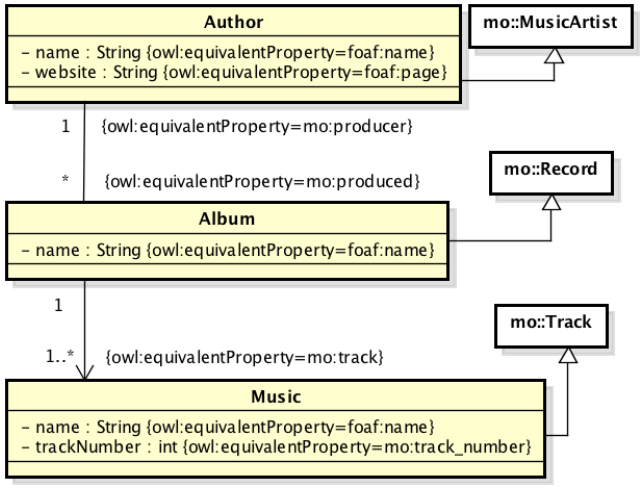

Figure 8: Domain Model for *SocialMusic*.

In all, 10 reports were produced, from which we illustrate two: *Medic*, in Figure 7, a WIS for keeping track of patients, their doctor's appointments, diagnostics and prescriptions[13]; and *SocialMusic*, in Figure 8, a WIS in which people can declare their musical interests for artists/bands, their albums or specific tracks. All course projects are available at an online version control repository.[14] Other than the aforementioned FOAF, *Medic* connected to the U.S. National Institutes of Health's Medical Subject Headings[15] vocabulary, whereas *SocialMusic* used the Music Ontology[16].

This experience allowed us to informally evaluate *FrameWeb-LD* by getting feedback from developers regarding the approach's usefulness, ease of use, completeness, etc. We recognize, however, that larger, more comprehensive and systematic experiments are necessary to properly evaluate our proposal.

[13]We realize this is not the kind of data one should publish as linked data. The course project has educational purposes only.
[14]https://github.com/orgs/dwws-ufes/
[15]http://www.nlm.nih.gov/mesh/
[16]http://musicontology.com/specification/

5. RELATED WORK

There is a very large body of work on linked data (LD) and the Semantic Web. We are interested, however, on those that, like us, propose an approach for integrating data from a Web-based Information System (WIS) into the *Web of Data*, i.e., connect it with well-known LD sets/vocabularies.

Hera [16] is a design method for Semantic WIS. It is focused on information systems that use Internet technologies for retrieving information from different sources on the Web and delivering it to users based on user preferences. It proposes an architecture with three layers: Semantic (specifies the data contents in terms of a conceptual model), Application (specifies the hypermedia view, representing navigation structures and user adaptations) and Presentation (details needed for producing the view in a concrete platform). Although somewhat similar, *FrameWeb-LD* proposes an architecture based on the use of well-established frameworks. Moreover, we propose the use of well-founded ontologies for describing content, whereas Hera conceptual models are based on operational ontologies (OWL, RDF(S)).

OntoWeaver [17] is an ontology-oriented approach for creating and maintaining personalized Web applications, i.e., whose contents are presented according to the need and preferences of its users and the kind of device being used to access the application. The declarative nature of the Web application specification allows a designer to manage and maintain it at the conceptual level. The internal knowledge model of OntoWeaver is frame-based and compatible with OCML, whereas we propose the use of an ontologically well-founded language based on UML, a more well-known modeling language for the average developer.

JOINT [14] is a Toolkit that supports the development of ontology-based applications through the integration of RDF and Object Oriented technologies. JOINT proposes the use of a triplestore instead of a relational database, generating code that integrates the RDF data into a Java application via a Knowledge Access Object (analogous to a Data Access Object [1], used by *FrameWeb*). Our proposal keeps the relational database, which is a popular choice for the architecture of WISs, adding a relational-to-RDF mapping layer on top of it in order to augment it with LD features.

Other publications also propose methods for building Semantic WISs, but focusing on specific concerns, such as multimedia [18], semantic portals [23] or integration of Web APIs [11]. Our work focuses on the use of ontologies and frameworks, supporting the publication of LD in RDF and describing semantic Web services.

6. CONCLUSIONS

In this paper, we presented *FrameWeb-LD*, a method for the integration of WIS into the Semantic Web based on *FrameWeb*. Using a systematic process and a well-founded language for building ontologies, the proposed approach generates linked data (LD) and semantic Web services for Web Information Systems (WIS) based on design models.

The idea behind our proposal is to decrease the burden on developers for publishing LD about their WIS and, thus,

promote the adoption of LD technologies. This is done by taking the specification of the data schema to a higher level of abstraction (design models) and automating most of the steps necessary for the generation of LD based on it.

This research is on-going work, with many limitations that should be addressed in future work, such as: (a) the *ReMaT* tool needs to be developed further to include the automation of more steps of the process (e.g., integrating Semantic Web Service descriptions with the vocabulary of the WIS); (b) the entire approach needs additional experiments to evaluate its usefulness, feasibility, ease of use, etc.; (c) OntoUML models currently include a lot of constraints on the generated OWL operational ontology due to its well-founded nature, with the purpose of guaranteeing consistency. Implementing in practice such constraints on the WIS is a very challenging task that needs further investigation.

We also intend to investigate ways to (semi-)automatically discover relevant vocabularies for a WIS being developed, helping developers make more and better connections between the vocabulary of the WIS and external ones from the *Web of Data*.

7. ACKNOWLEDGMENTS

Nemo (http://nemo.inf.ufes.br) is currently supported by Brazilian research agencies Fapes (# 0969/2015), CNPq (# 485368/2013-7, # 461777/2014-2), and by Ufes' FAP (# 6166/2015). The authors would like to thank Bruno Borlini Duarte for his contributions to the OntoUML model of C2D and all the students that contributed to *FrameWeb-LD*'s evaluation.

8. REFERENCES

[1] D. Alur, J. Crupi, and D. Malks. *Core J2EE Patterns: Best Practices and Design Strategies*. Prentice Hall / Sun Microsystems Press, 2nd edition, 2003.

[2] T. Berners-Lee. Linked Data - Design Issues, http://www.w3.org/DesignIssues/LinkedData.html (last access: May 7th, 2015), 2006.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[4] L. DeMichiel and B. Shannon. JSR 342: Java^{TM}Platform, Enterprise Edition 7 (Java EE 7) Specification, https://jcp.org/en/jsr/detail?id=342 (last access: April 29th, 2015).

[5] D. Elenius, G. Denker, D. Martin, F. Gilham, J. Khouri, S. Sadaati, and R. Senanayake. The OWL-S editor–a development tool for semantic web services. In *The Semantic Web: Research and Applications*, pages 78–92. Springer, 2005.

[6] R. A. Falbo. SABiO: Systematic Approach for Building Ontologies. In G. Guizzardi, O. Pastor, Y. Wand, S. de Cesare, F. Gailly, M. Lycett, and C. Partridge, editors, *Proc. of the Proceedings of the 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering*. CEUR, sep 2014.

[7] R. A. Falbo, G. Guizzardi, and K. C. Duarte. An ontological approach to domain engineering. In *Proc. of the 14th International Conference on Software Engineering and Knowledge Engineering*, pages 351–358. ACM, jul 2002.

[8] M. Grüninger and M. S. Fox. Methodology for the Design and Evaluation of Ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing*, apr 1995.

[9] N. Guarino, D. Oberle, and S. Staab. What is an Ontology? In S. Staab and R. Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 1–17. Springer, 2 edition, 2009.

[10] G. Guizzardi. *Ontological Foundations for Structural Conceptual Models*. Phd thesis, University of Twente, The Netherlands, 2005.

[11] M. Hausenblas. Exploiting linked data to build web applications. *IEEE Internet Computing*, 13(4):68, 2009.

[12] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers, 2011.

[13] P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. CRC Press, 2009.

[14] O. Holanda, S. Isotani, I. I. Bittencourt, E. Elias, and T. Tenório. JOINT: Java ontology integrated toolkit. *Expert Systems with Applications*, 40(16):6469–6477, 2013.

[15] I. Horrocks. Ontologies and the Semantic Web. *Communications of the ACM*, 51(12):58–67, 2008.

[16] G.-J. Houben, P. Barna, F. Frasincar, and R. Vdovjak. Hera: Development of Semantic Web information systems. In *Web Engineering*, pages 529–538. Springer, 2003.

[17] Y. Lei, E. Motta, and J. Domingue. OntoWeaver: An ontology-based approach to the design of data-intensive web sites. *Journal of Web Engineering*, 4(3):244, 2005.

[18] F. Lima and D. Schwabe. Modeling applications for the Semantic Web. In *Web Engineering*, pages 417–426. Springer, 2003.

[19] B. F. Martins and V. E. S. Souza. A Model-Driven Approach for the Design of Web Information Systems based on Frameworks. In *Proc. of the 21st Brazilian Symposium on Multimedia and the Web*, pages 41–48. ACM, oct 2015.

[20] S. A. McIlraith, T. C. Son, and H. Zeng. Semantic Web Services. *IEEE intelligent systems*, 16(2):46–53, 2001.

[21] D. Schmidt, M. Stal, H. Rohnert, and F. Buschmann. *Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects*. Wiley, 2013.

[22] V. E. S. Souza, R. A. Falbo, and G. Guizzardi. Designing Web Information Systems for a Framework-based Construction. In T. Halpin, E. Proper, and J. Krogstie, editors, *Innovations in Information Systems Modeling: Methods and Best Practices*, chapter 11, pages 203–237. IGI Global, 1 edition, 2009.

[23] N. Stojanovic, A. Maedche, S. Staab, R. Studer, and Y. Sure. SEAL: a framework for developing SEmantic PortALs. In *Proceedings of the 1st International Conference on Knowledge Capture*, pages 155–162. ACM, 2001.