τOWL-Manager: A Tool for Managing Temporal Semantic Web Documents in the τOWL Framework

Abir Zekri, Zouhaier Brahmia
University of Sfax
Sfax, Tunisia
emails: abir.zekri@fsegs.rnu.tn, zouhaier.brahmia@fsegs.rnu.tn

Fabio Grandi
University of Bologna
Bologna, Italy
email: fabio.grandi@unibo.it

Rafik Bouaziz
University of Sfax
Sfax, Tunisia
email: raf.bouaziz@fsegs.rnu.tn

***Abstract*—Several semantic web-based applications (e.g., e-commerce, e-government and e-health applications) require temporal versioning of ontology instances, in order to represent, store and retrieve time-varying ontologies. However, commercial systems do not provide any support for creating and updating temporal ontologies. In this paper, we propose a prototype system, named Temporal OWL 2 Web Ontology Language Manager (τOWL-Manager), which implements our τOWL framework and supports temporal versioning of ontology instances. It allows (i) creating and validating a temporal semantic web document, by augmenting an OWL 2 ontology schema with a set of logical and physical annotations, and (ii) creating and maintaining time-varying ontology instance documents, by generating a new timestamped version of each ontology instance document when updates are applied.**

Keywords–Semantic Web; Ontology; OWL 2; τXSchema; Logical annotations; Physical annotations; Temporal database; XML Schema; XML

I. INTRODUCTION

Due to the dynamic nature of the Web, ontologies [2]—like other components of the Web 3.0 including databases and Web pages—evolve over time to reflect and model changes occurring in the real-world. Furthermore, several Semantic Web-based applications (like e-commerce, e-government and e-health applications) require keeping track of ontology evolution and versioning with respect to time, in order to represent, store and retrieve time-varying ontologies.

Unfortunately, while there is a sustained interest for temporal and evolution aspects in the research community [3], existing Semantic Web [4] standards, state-of-the-art ontology editors and knowledge representation tools do not provide any built-in support for managing temporal ontologies. In particular, the W3C OWL 2 recommendation [5][6] lacks explicit support for time-varying ontologies, at both schema and instance levels. Thus, a Knowledge Base Administrator (KBA), i.e., a knowledge engineer or a maintainer of semantics-based Web resources, must use ad hoc techniques when there is a need, for example, to specify an OWL 2 ontology schema for time-varying ontology instances.

On the other hand, in order to handle temporal ontology evolution in an effective and systematic manner and to allow historical queries to be efficiently executed on time-varying ontologies, a built-in temporal ontology management system is needed. For that purpose, we proposed in our previous work [1] a framework, called τOWL, for managing temporal Semantic Web documents, through the use of a temporal OWL 2 extension. In fact, we want to introduce with τOWL a principled and systematic approach to the temporal extension of OWL 2, similar to that Snodgrass and colleagues did with their Temporal XML Schema (τXSchema) [7][8] to the eXtensible Markup Language (XML) and XML Schema [9]. τXSchema is a powerful framework (i.e., a data model equipped with a suite of tools) for managing temporal XML documents, well known in the database research community and, in particular, in the field of temporal XML [10]. Moreover, in the previous work [11], with the aim of completing the framework, we augmented τXSchema by defining necessary schema change operations.

Being defined as a τXSchema-like framework, τOWL allows creating a temporal OWL 2 ontology from a conventional (i.e., non-temporal) OWL 2 ontology specification and a set of logical (or temporal) and physical annotations. Logical annotations identify which components of a Semantic Web document can vary over time; physical annotations specify how the time-varying aspects are represented in the document. By using temporal schema and annotations to introduce temporal aspects in the conventional Semantic Web, our framework (i) guarantees logical and physical data independence [12] for temporal ontologies and (ii) provides a low-impact solution since it requires neither modifications of existing Semantic Web documents nor extensions to the OWL 2 recommendation and Semantic Web standards.

Furthermore, while there is a lot of research works on managing temporal ontologies [13][14][15][16], only two research tools have been proposed to handle some particular aspects: Stock Recommendations Aggregation System (SRAS) [17], which is centered around the aggregation of stock recommendations and financial data, and CHRONOS [18], which is a reasoner over temporal information in OWL ontologies. Current commercial solutions in the Semantic Web area (Oracle Semantic Technology [19], IBM Scalable Ontology Repository (SOR) [20], and IBM DB2 Resource Description Framework (RDF) [21]) do not include features for supporting time in ontologies.

In order to (i) show the feasibility of our τOWL approach [1], (ii) facilitate a KBA when he/she has to create a temporal ontology and manipulate its instances, and (iii)

 ISBN: 978-1-61208-420-6

fill the lack of support noticed in commercial knowledge management systems, we propose in this paper a prototype system, named τOWL-Manager, which allows a KBA (i) to create and validate τOWL ontology schemata, and (ii) to create and update τOWL ontology instance documents. When modified, instance documents are augmented with timestamps to support temporal versioning.

With regard to our previous work [1], the current one focuses on implementing our τOWL framework; the result, τOWL-Manager, could be a first step towards providing commercial support for temporal ontologies.

The remainder of the paper is organized as follows. Section II describes our τOWL framework, previously proposed in [1]: the architecture of τOWL is presented and details on all its components and support tools are given. Section III illustrates the use of τOWL through an example. Section IV proposes our prototype tool, τOWL-Manager: its architecture and some screenshots showing its functioning are provided. Section V provides a summary of the paper and some remarks about our future work.

II. THE τOWL FRAMEWORK

In this section, we present our τOWL framework for handling temporal Semantic Web documents. We describe the overall architecture of τOWL. Since τOWL is a τXSchema-like framework, we were inspired by the τXSchema architecture and tools while defining the architecture and tools of τOWL. More details on our framework can be found in [1] and [22].

The τOWL framework allows a KBA to create a temporal OWL 2 schema for temporal OWL 2 instances from a conventional OWL 2 schema, logical annotations, and physical annotations. Since it is a τXSchema-like framework, τOWL use the following principles: separation between (i) the conventional (i.e., non-temporal) schema and the temporal schema, and (ii) the conventional instances and the temporal instances; (iii) use of logical and physical annotations to specify temporal and physical aspects, respectively, at schema level.

Figure 1 illustrates the architecture of τOWL. The framework is based on the OWL 2 language [5][6], which is a W3C standard ontology language for the Semantic Web. It allows defining both schema (i.e., entities, axioms, and expressions) and instances (i.e., individuals) of ontologies.

The KBA starts by creating the *conventional schema* (box 7), which is an OWL 2 ontology that models the concepts of a particular domain and the relations between these concepts, without any temporal aspect. To each conventional schema corresponds a set of conventional OWL 2 instances (box 12). As recommended in the the OWL 2 specification [6], τOWL deals with OWL 2 ontologies with an RDF/XML syntax [23].

After that, the KBA augments the conventional schema with *logical* and *physical annotations*, which allow him/her to express, in an explicit way, all requirements dealing with the representation and the management of temporal aspects associated to the components of the conventional schema, as described in the following.

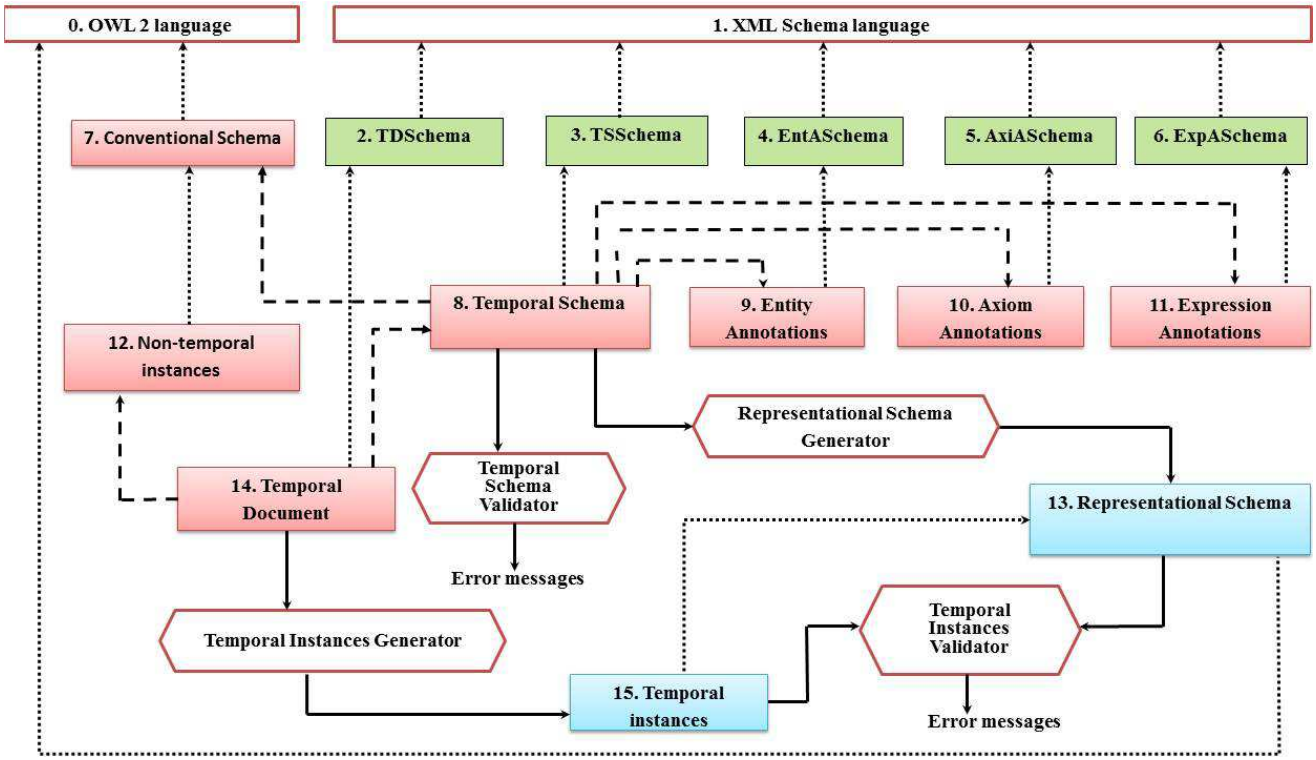

Figure 1. Overall architecture of τOWL.

Logical annotations [8] allow the KBA to specify (i) whether a conventional schema component varies over valid time and/or transaction time, (ii) whether its lifetime is described as a continuous state or a single event, (iii) whether the component may appear at certain times (and not at others), and (iv) whether its content changes.

Physical annotations [8] allow the KBA to specify the timestamp representation options chosen, such as where the timestamps are placed and their kind (i.e., valid time or transaction time) and the kind of representation adopted. Timestamps can be located either on time-varying components (as specified by the logical annotations) or somewhere above such components. Two OWL 2 documents with the same logical information will look very different if we change the location of their physical timestamps.

Finally, when the KBA finishes annotating the conventional schema and asks the system to save his/her work, this latter creates the *temporal schema* (box 8) in order to provide the linking information between the conventional schema and its corresponding logical and physical annotations. The temporal schema is a standard XML document which ties the conventional schema, the entity annotations, the axiom annotations, and the expression annotations together. In the τOWL framework, the temporal schema is the logical equivalent of the conventional OWL 2 schema in a non-temporal context. This document contains sub-elements that associate a series of conventional schema definitions with entity annotations, axiom annotations, and expression annotations, along with the time span during which the association was in effect. The schema for the temporal schema document is the XML Schema Definition document *TSSchema* (box 3).

To complete the picture, after creating the temporal schema, the system creates a *temporal document* (box 14) in order to link each conventional ontology instance document (box 12), which is valid to a conventional ontology schema (box 7), to its corresponding temporal ontology schema (box 8), and more precisely to its corresponding logical and physical annotations (which are referenced by the temporal schema). A temporal document is a standard XML document that maintains the evolution of a non-temporal ontology instance document over time, by recording all of the versions (or temporal slices) of the document with their corresponding timestamps and by specifying the temporal schema associated to these versions. This document contains sub-elements that associate a series of conventional ontology instance documents with logical and physical annotations (on entities, axioms, and expressions), along with the time span during which the association was in effect. Thus, the temporal document is very important for making easy the support of temporal queries working on past versions or dealing with changes between versions. The schema for the temporal document is the XML Schema Definition document *TDSchema* (box 2).

III. ILLUSTRATIVE EXAMPLE

In order to show the functioning of the τOWL approach and how management of temporal ontology document versions is dealt with in it, we provide an example concerning the evolution of an ontology based on Friend Of A Friend (FOAF). The FOAF [24] project is creating a Web of machine-readable pages describing people, the links between them and the things they create and do.

Suppose that a Web site "Society-Web" publishes the FOAF definition for their users and that the webmaster of this Web site wants to keep track of the changes performed on FOAF RDF [25] information. We will focus in this example on one user whose name is "Khalid Sinan".

Suppose that on January 15, 2014, the KBA creates a conventional ontology schema, named "PersonSchema_V1.owl" (Figure 2), and a conventional ontology instance document, named "Persons_V1.rdf" (Figure 3), which is valid with respect to this schema. We assume that the KBA defines also a set of logical and physical annotations, associated to that conventional schema; they are stored in an ontology annotation document titled "PersonAnnotations_V1.xml" as shown in Figure 4.

Notice that the conventional (i.e., non-temporal) schema (Figure 1) for the FOAF RDF document (Figure 2) is the schema for an individual version, which allows updating and querying individual versions. The conventional ontology instance document describes, according to the FOAF ontology, the personal information of "Khalid Sinan" (i.e., name and nickname) and the information about his online accounts on diverse sites (i.e., the home page of the site, and the account name of the user). In this example, we only consider the user account on the "Facebook" Web site.

```
<rdf:RDF>
  <owl:Ontology  rdf:about="http://purl.org/
                            az/foaf#">
    <rdfs:Class  rdf:about="#Person">
      <rdf:type  rdf:resource="http://www.w3.org/
                               2002/07/owl#Class"/>
    </rdfs:Class>
    <rdf:Property  rdf:about="#holdsAccount">
      <rdf:type  rdf:resource="http://www.w3.org/
                    2002/07/owl#ObjectProperty"/>
      <rdfs:domain  rdf:resource="#Person"/>
      <rdfs:range  rdf:resource="#OnlineAccount"/>
    </rdf:Property>
    <rdf:Property  rdf:about="#accountName">
      <rdf:type  rdf:resource="http://www.w3.org/
                  2002/07/owl#DatatypeProperty"/>
      <rdfs:domain rdf:resource="#OnlineAccount"/>
    </rdf:Property>
    …
</rdf:RDF>
```

Figure 2. An RDF/XML extract from the OWL 2 FOAF ontology.

```
…
<foaf:Person rdf:ID="#Person1">
  <foaf:name>Khalid Sinan</foaf:name>
  <foaf:nick>Khal</foaf:nick>
  <foaf:holdsAccount>
    <foaf:OnlineAccount rdf:about="
          https://www.facebook.com/Khalid.Sinan">
      <foaf:accountName>Khal_Sinan
      </foaf:accountName>
    </foaf:OnlineAccount>
  </foaf:holdsAccount>
</foaf:Person>
…
```

Figure 3. A fragment of Khalid FOAF RDF document on January 15, 2014.

 ISBN: 978-1-61208-420-6

```
<?xml version="1.0" encoding="UTF-8"?>
<ontologyAnnotationSet>
  <logicalAnnotations>
    <item target="/Person/nick">
      <validTime kind="state" content="varying"
                 existence="constant"/>
    </item>
  </logicalAnnotations>
  <physicalAnnotations>
    <stamp target="Person/nick"
           dataInclusion="expandedVersion">
      <stampkind timeDimension="validTime"
                 stampBounds="extent"/>
    </stamp>
  </physicalAnnotations>
</ontologyAnnotationSet>
```

Figure 4. The annotation document on January 15, 2014.

After that, the system creates the temporal ontology schema in Figure 5 (that ties "PersonSchema_V1.owl" and "PersonAnnotations_V1.xml" together), which is stored in an XML file named "PersonTemporalSchema.xml". Consequently, the system uses the temporal ontology schema of Figure 5 and the conventional ontology document in Figure 3 to create a temporal document as in Figure 6, that lists both versions (i.e., temporal "slices") of the conventional ontology documents with their associated timestamps. The squashed version of this temporal document, which could be generated by the Temporal Instances Generator, is provided in Figure 7.

```
<?xml version="1.0" encoding="UTF-8"?>
<temporalOntologySchema>
  <conventionalOntologySchema>
    <sliceSequence>
      <slice location="PersonSchema_V1.owl"
             begin="2014-01-15" />
    </sliceSequence>
  </conventionalOntologySchema>
  <ontologyAnnotationSet>
    <sliceSequence>
      <slice location="PersonAnnotations_V1.xml"
             begin="2014-01-15" />
    </sliceSequence>
  </ontologyAnnotationSet>
</temporalOntologySchema>
```

Figure 5. The temporal schema on January 15, 2014.

```
<?xml version="1.0" encoding="UTF-8"?>
<td:temporalRoot temporalSchemaLocation=
                   "PersonTemporalSchema.xml"/>
  <td:sliceSequence>
    <td:slice location="Persons_V1.rdf"
              begin="2014-01-15" />
  </td:sliceSequence>
</td:temporalRoot>
```

Figure 6. The temporal document on January 15, 2014.

On February 08, 2014, Khalid modified his nickname from "Khal" to "Elkhal" and his account name of Facebook from "Khal_Sinan" to "Elkhal_Sinan". Thus, the system updates the conventional ontology document "Persons_V1.rdf" to produce a new conventional ontology document named "Persons_V2.rdf" (Figure 8). Since the conventional ontology schema (i.e., PersonSchema_V1.owl) and the ontology annotation document (i.e., PersonAnnotations_V1.xml) are not changed, the temporal ontology schema (i.e., PersonTemporalSchema.xml) is consequently not updated. However, the Temporal Instances Generator tool updates the temporal document, in order to include the new slice of the conventional ontology document, as shown in Figure 9. The squashed version of the updated temporal document is provided in Figure 10.

Obviously, each one of the squashed documents (Figure 7 and Figure 10) must conform to a particular schema, that is the representational schema, which is generated by the Representational Schema Generator from the temporal schema shown in Figure 5.

```
<foaf:Person rdf:ID="#Person1">
  <foaf:name>Khalid Sinan</foaf:name>
  <nick_RepItem>
    <nick_Version>
      <timestamp_ValidExtent begin="2014-01-15"
                             end="now" />
      <foaf:nick>Khal</foaf:nick>
    </nick_Version>
  </nick_RepItem>
  <foaf:holdsAccount>
    <foaf:OnlineAccount rdf:about="
          https://www.facebook.com/Khalid.Sinan">
      <accountName_RepItem>
        <accountName_Version>
          <timestamp_ValidExtent
                 begin="2014-01-15" end="now" />
          <foaf:accountName>Khal_Sinan
          </foaf:accountName>
        </accountName_Version>
      </accountName_RepItem>
    </foaf:OnlineAccount>
  </foaf:holdsAccount>
</foaf:Person>
```

Figure 7. The squashed document correponding to the temporal document on January 15, 2014.

```
…
<foaf:Person rdf:ID="#Person1">
  <foaf:name>Khalid Sinan</foaf:name>
  <foaf:nick>Elkhal</foaf:nick>
  <foaf:holdsAccount>
    <foaf:OnlineAccount rdf:about="
          https://www.facebook.com/Khalid.Sinan">
      <foaf:accountName>Elkhal_Sinan
      </foaf:accountName>
    </foaf:OnlineAccount>
  </foaf:holdsAccount>
</foaf:Person>
…
```

Figure 8. A fragment of Khalid FOAF RDF document on February 08, 2014.

```
<?xml version="1.0" encoding="UTF-8"?>
<td:temporalRoot temporalSchemaLocation=
                   "PersonTemporalSchema.xml"/>
  <td:sliceSequence>
    <td:slice location="Persons_V1.rdf"
              begin="2014-01-15" />
    <td:slice location="Persons_V2.rdf"
              begin="2014-02-08" />
  </td:sliceSequence>
</td:temporalRoot>
```

Figure 9. The temporal document on February 08, 2014.

 ISBN: 978-1-61208-420-6

```
<foaf:Person rdf:ID="#Person1">
  <foaf:name>Khalid Sinan</foaf:name>
  <nick_RepItem>
    <nick_Version>
      <timestamp_ValidExtent begin="2014-01-15"
                             end="2014-02-07" />
      <foaf:nick>Khal</foaf:nick>
    </nick_Version>
    <nick_Version>
      <timestamp_ValidExtent begin="2014-02-08"
                             end="now" />
      <foaf:nick>Elkhal</foaf:nick>
    </nick_Version>
  </nick_RepItem>
  <foaf:holdsAccount>
    <foaf:OnlineAccount rdf:about="
          https://www.facebook.com/Khalid.Sinan">
      <accountName_RepItem>
        <accountName_Version>
          <timestamp_ValidExtent begin="2014-01-15"
                                 end="2014-02-07"/>
          <foaf:accountName>Khal_Sinan
          </foaf:accountName>
        </accountName_Version>
        <accountName_Version>
          <timestamp_ValidExtent begin="2014-02-08"
                                 end="now" />
          <foaf:accountName>Elkhal_Sinan
          </foaf:accountName>
        </accountName_Version>
      </accountName_RepItem>
    </foaf:OnlineAccount>
  </foaf:holdsAccount>
</foaf:Person>
```

Figure 10. The squashed document correponding to the temporal document on February 08, 2014.

IV. IMPLEMENTATION

In this section, we describe a prototype system, named τOWL-Manager, which implements our τOWL approach and shows its feasibility. It allows (i) the specification and validation of τOWL ontologies schemata, and (ii) the creation and maintenance of τOWL ontology instance documents. Each update operation on an instance document gives rise to a new version of this document with its corresponding timestamp.

τOWL-Manager is a Java (JDK 1.7) application, developed in the Integrated Development Environment (IDE) "Eclipse Helios", using (i) the OWL Application Programming Interface (API) [26], which is a Java API and a reference implementation, for creating and manipulating OWL ontologies, and (ii) the Java Document Object Model (JDOM) API for creating and manipulating XML files. In the following, we first describe the architecture of τOWL-Manager and then provide some screenshots showing its use. Notice that these screenshots deal with the same example presented in Section III.

A. *Architecture of τOWL-Manager*

The overall architecture of τOWL-Manager is depicted in Figure 11. It is composed of three layers: presentation layer, business layer, and storage layer.

The **presentation layer** includes an interface for constructing temporal ontologies and an interface for creating and updating ontology instances.

The **business layer** contains two modules: one for managing temporal ontologies, named "Temporal Ontology Manager", and the other for managing ontology instances, named "Ontology Instance Document Manager". The "Temporal Ontology Manager" first generates the files corresponding to the temporal ontology schema, that is the conventional schema file and the annotation document file, from the specifications expressed by the KBA in its interface. Then, it checks the validity of the generated files and creates the temporal schema file, which ties together the two other files.

The **storage layer** contains the repository of resources making up temporal ontologies and associated instances, named τOWL Repository.

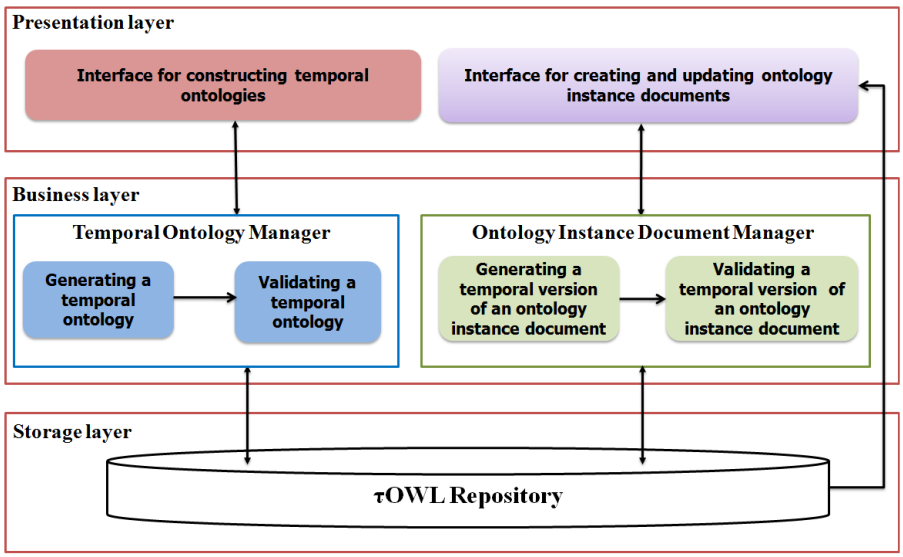

Figure 11. Architecture of τOWL-Manager.

B. *Screenshots of τOWL-Manager*

Currently, τOWL-Manager allows a KBA to perform two activities: (i) creating and validating temporal ontologies, and (ii) creating and updating ontology instances. In the following, we illustrate its functioning and show its use for each one of the two activities, via the example of Section III.

1) Constructing and validating temporal ontologies

To construct a new temporal ontology, the KBA has to perform the following tasks:

i) He/she starts by creating a τOWL project. To this aim, the KBA has to provide a reference to an existing valid conventional ontology schema (definition of an ontology schema from scratch is not supported in the current version of τOWL-Manager). Assume here that the KBA has chosen the FOAF ontology.

ii) After that, the KBA annotates the new conventional ontology schema by some logical and physical annotations. Figure 12 shows the specification of some logical annotations on the class "Person" and Figure 13 shows the specification of some physical annotations on the same class.

Notice that a τOWL project is a set of folders:

- Annotations: it contains the file corresponding to the logical and physical annotation document of a τOWL ontology;
- Conventional Ontology Instance Documents: it stores all the versions of conventional ontology instance documents;
- Conventional Ontology Schema: it includes the conventional ontology schema file of a τOWL ontology;
- Representational Schema: it stores the representational schema file;
- Temporal Document: it includes the temporal document (which is generated automatically);
- Temporal Ontology Instance Documents: it contains all the versions of temporal ontology instance documents.
- Temporal Schema: it contains the temporal schema file.

2) Creating and versioning ontology instance documents

We show in Figure 14 an ontology instantiation. After the KBA has chosen a τOWL ontology schema, he/she can create its instances (Figure 14). Finally, he/she should save his/her work, through the "Save" button. Consequently, the system generates an RDF file corresponding to the conventional ontology instances which have been created. Such a file is generated using the OWL API and validated using the Pellet reasoner. Furthermore, the system automatically updates the temporal document in order to add a new slice corresponding to the new version of the ontology instance document.

Moreover, τOWL-Manager allows keeping track of ontology instances when they evolve over time. Figures 15 and 16 show an example of maintaining the history of an ontology instance evolution: first the KBA chooses "Persons_V1.rdf" as the ontology instance document version that must be updated (Figure 15).

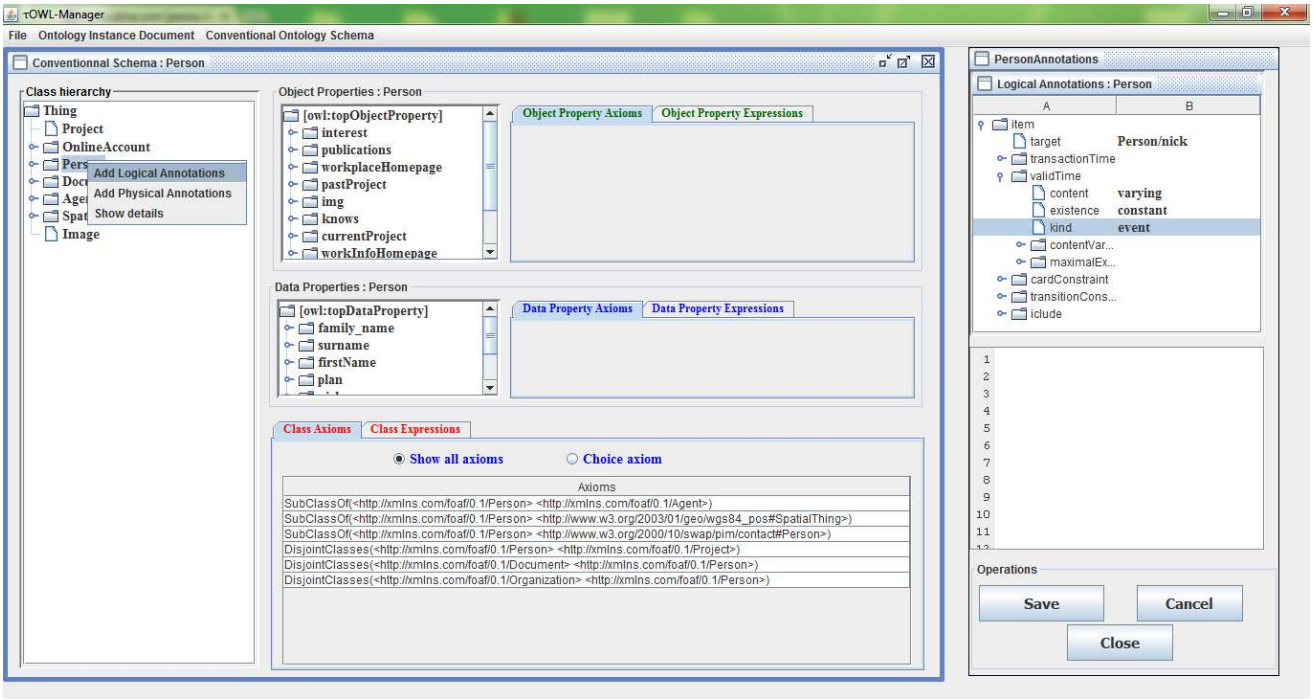

Figure 12. Specifying some logical annotations on the conventional ontology schema.

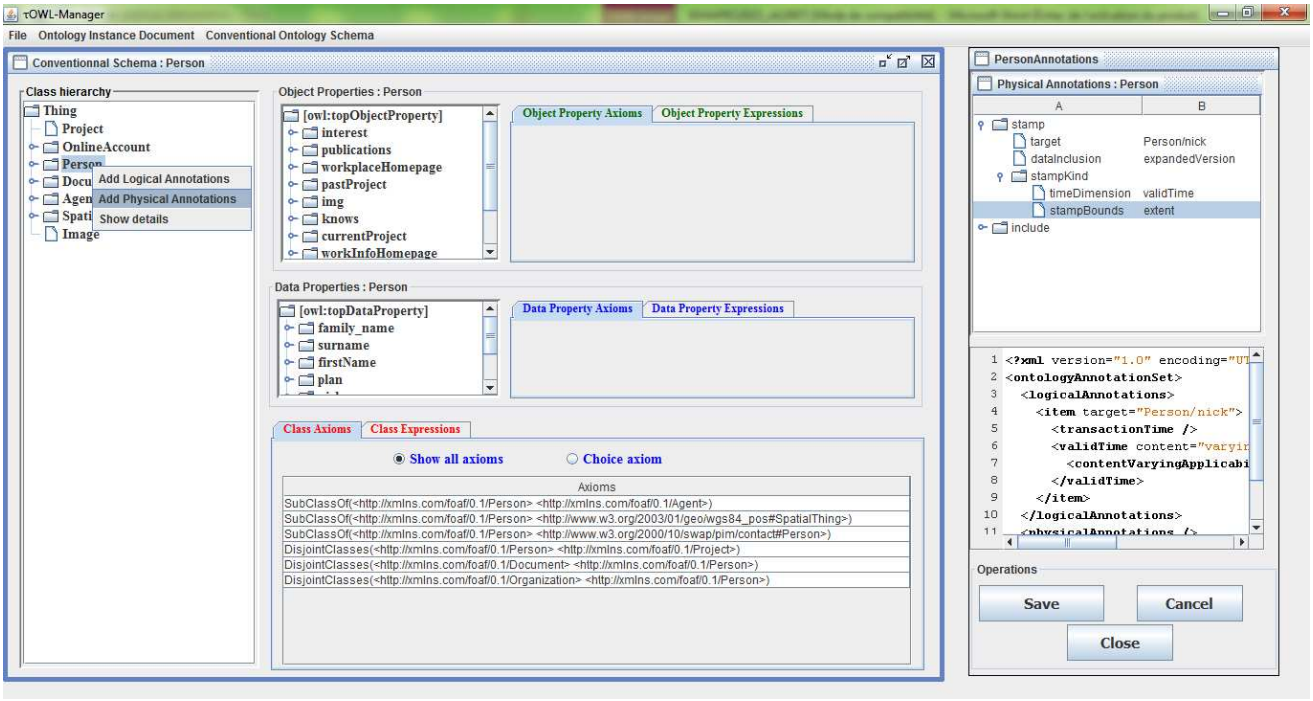

Figure 13. Specifying some physical annotations on the conventional ontology schema.

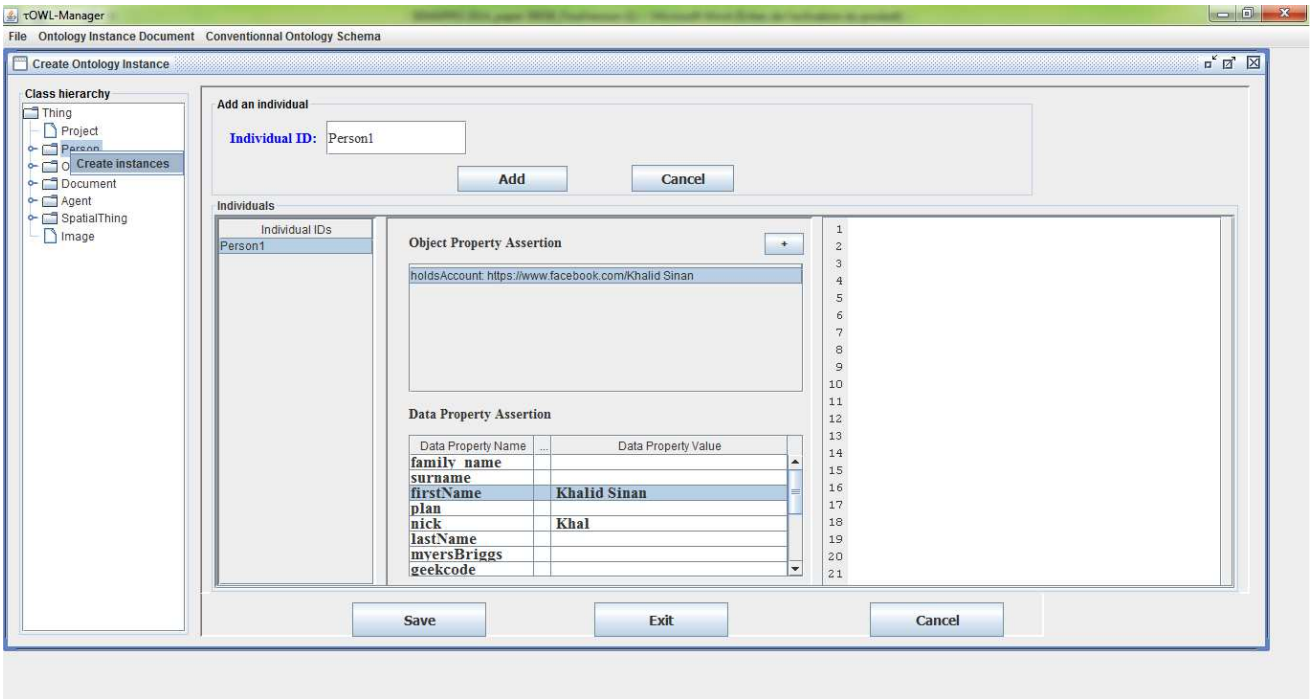

Figure 14. Populating a conventional ontology.

 ISBN: 978-1-61208-420-6

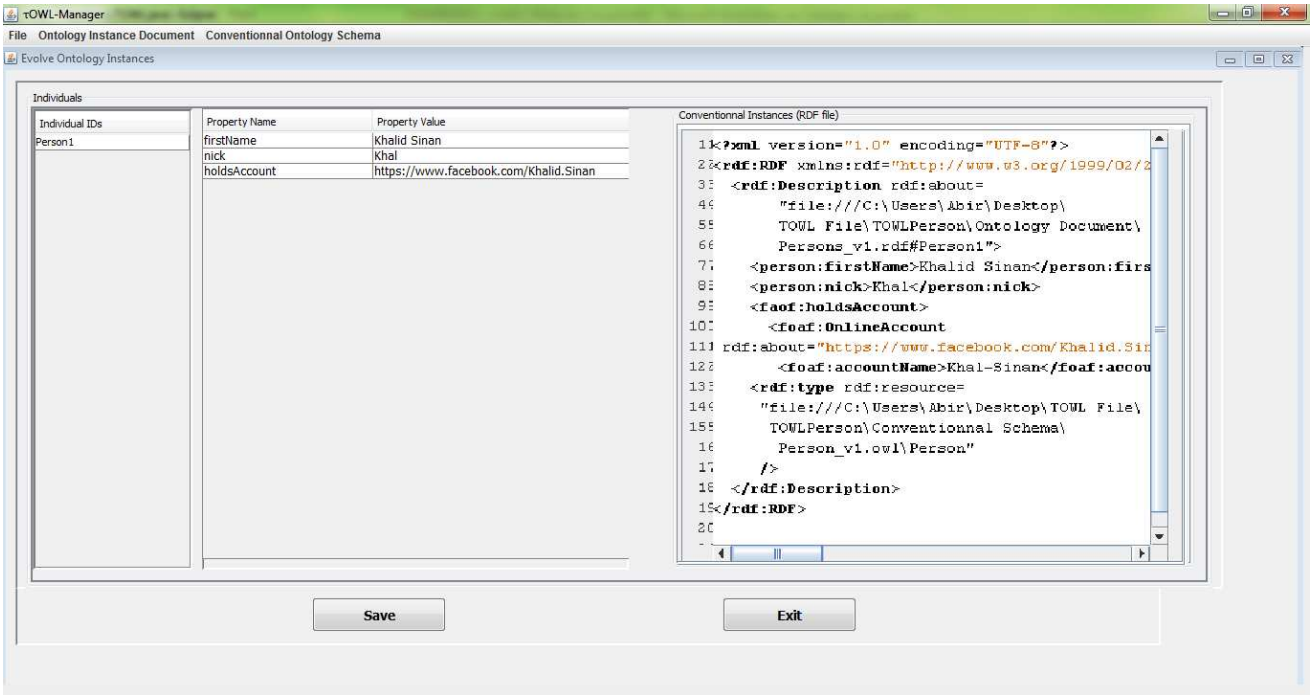

Figure 15. Showing the chosen conventional ontology instance document version.

Figure 16 shows that the KBA has modified the chosen ontology instance document version (by modifying the nick and the account name of the Person "Khalid Sinan"). Thus, the system automatically generates a new version of the ontology instance document. After verifying that the new ontology instance document version is different from its predecessor, the system adds it to the folder "Conventional Ontology Instance Documents" of the τOWL project. Moreover, the creation of a new version of an ontology instance document causes an automatic update of the temporal document.

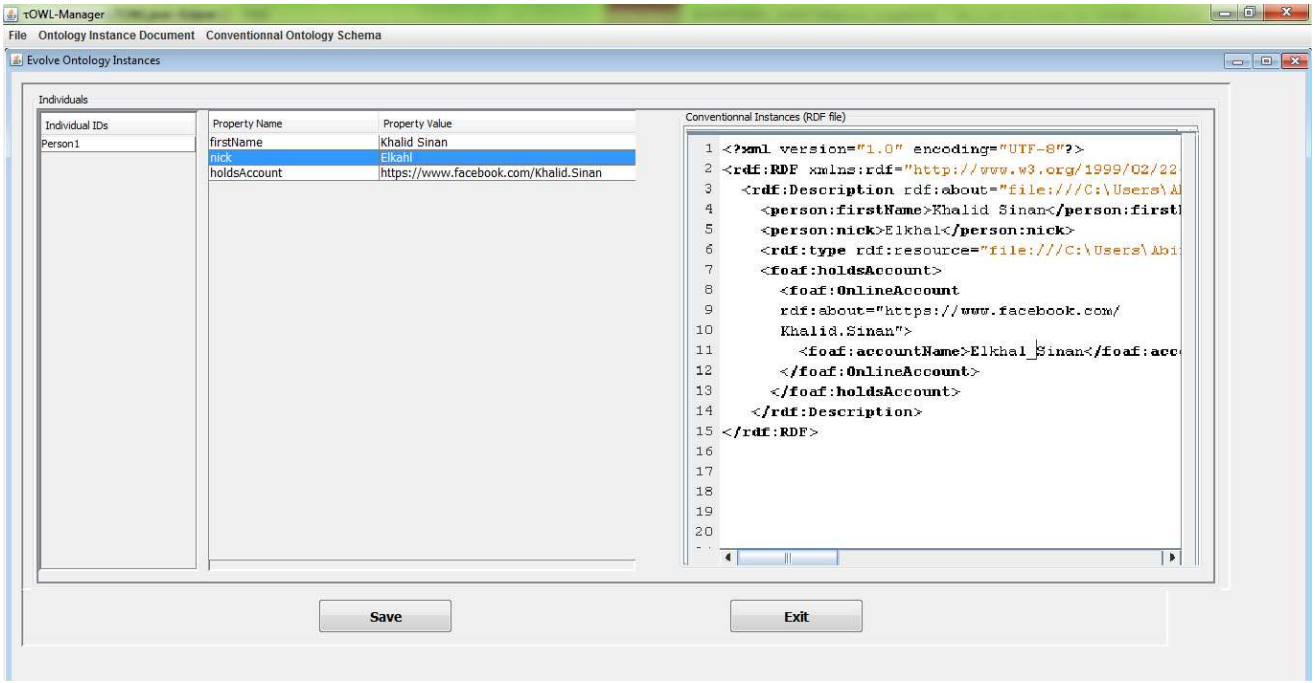

Figure 16. Updating the chosen conventional ontology instance document version (changing the nick and the account name of Khalid).

 ISBN: 978-1-61208-420-6

V. CONCLUSION AND FUTURE WORK

In this paper, we have presented τOWL-Manager, a prototype tool for specifying temporal ontologies and temporal instance versioning, in the τOWL framework, demonstrating its feasibility. It helps a KBA to create temporal ontologies and manipulate its instances, overcoming the lack of support detected in state-of-the-art commercial knowledge management systems and research tools. Thus, it could be considered as a first step towards providing commercial support for temporal ontologies.

Our future work aims at extending τOWL-Manager to also support temporal versioning of the schema itself, in the τOWL framework. Such extension requires, as a first step, the definition of necessary schema change operations, that is operations acting on conventional schema, annotations and temporal schema. A subset of these operations has been defined in our recent work [27].

REFERENCES

[1] A. Zekri, Z. Brahmia, F. Grandi, and R. Bouaziz, "τOWL: A Framework for Managing Temporal Semantic Web Documents," Proceedings of the 8th International Conference on Advances in Semantic Processing (SEMAPRO 2014), Rome, Italy, 24-28 August 2014, pp. 33-41.

[2] N. Guarino (Ed.), Formal Ontology in Information Systems, IOS Press, Amsterdam, 1998.

[3] F. Grandi, "Introducing an Annotated Bibliography on Temporal and Evolution Aspects in the Semantic Web," SIGMOD Record, vol. 41, December 2012, pp. 18-21.

[4] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," Scientific American, vol. 284, May 2001, pp. 34-43.

[5] W3C, OWL 2 Web Ontology Language – Primer (Second Edition), W3C Recommendation, 11 December 2012. <http://www.w3.org/TR/owl2-primer/> [retrieved: June, 2015]

[6] W3C, OWL 2 Web Ontology Language – Document Overview (Second Edition), W3C Recommendation, 11 December 2012. <http://www.w3.org/TR/owl2-overview/> [retrieved: June, 2015]

[7] F. Currim, S. Currim, C. E. Dyreson, and R. T. Snodgrass, "A Tale of Two Schemas: Creating a Temporal XML Schema from a Snapshot Schema with tXSchema," Proceedings of EDBT'2004, Heraklion, Crete, Greece, 14-18 March 2004, pp. 348-365.

[8] R. T. Snodgrass, C. E. Dyreson, F. Currim, S. Currim, and S. Joshi, "Validating Quicksand: Schema Versioning in τXSchema," Data and Knowledge Engineering, vol. 65, May 2008, pp. 223-242.

[9] W3C, XML Schema Part 0: Primer Second Edition, W3C Recommendation, 28 October 2004. <http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/> [retrieved: June, 2015]

[10] C. E. Dyreson and F. Grandi, "Temporal XML," in L. Liu and M. T. Özsu (Eds.), Encyclopedia of Database Systems, Springer US, 2009, pp. 3032-3035.

[11] Z. Brahmia, F. Grandi, B. Oliboni, and R. Bouaziz, "Schema Change Operations for Full Support of Schema Versioning in the τXSchema Framework," International Journal of Information Technology and Web Engineering, vol. 9, April-June 2014, pp. 20-46.

[12] T. Burns et al., "Reference Model for DBMS Standardization, Database Architecture Framework Task Group (DAFTG) of the ANSI/X3/SPARC Database System Study Group," SIGMOD Record, vol. 15, March 1986, pp. 19-58.

[13] C. Gutiérrez, C. A. Hurtado, and A. A. Vaisman, "Introducing time into RDF," IEEE Transactions on Knowledge and Data Engineering, vol. 19, February 2007, pp. 207-218.

[14] F. Grandi and M. R. Scalas, "The valid ontology: A simple OWL temporal versioning framework," Proceedings of the 3rd International Conference on Advances in Semantic Processing (SEMAPRO 2009), Sliema, Malta, 11-16 October 2009, pp. 98-102.

[15] M. J. O'Connor and A. K. Das, "A method for representing and querying temporal information in OWL," In Biomedical Engineering Systems and Technologies, volume 127 of Communications in Computer and Information Science, pp. 97-110. Springer-Verlag, Heidelberg, Germany, 2011.

[16] V. Milea, F. Frasincar, and U. Kaymak, "tOWL: A Temporal Web Ontology Language," IEEE Transactions on Systems, Man, and Cybernetics, Part B, vol. 42, February 2012, pp. 268-281.

[17] V. Milea, F. Frasincar, and U. Kaymak, "Knowledge Engineering in a Temporal Semantic Web Context," Proceedings of the 8th International Conference on Web Engineering (ICWE 2008), Yorktown Heights, New York, USA, 14-18 July 2008, pp. 65-74.

[18] E. Anagnostopoulos, S. Batsakis, and E. G. M. Petrakis, "CHRONOS: A Reasoning Engine for Qualitative Temporal Information in OWL," Proceedings of the 17th International Conference in Knowledge-Based and Intelligent Information & Engineering Systems (KES 2013), Kitakyushu, Japan, 9-11 September 2013, pp. 70-77.

[19] Z. Wu et al., "Implementing an Inference Engine for RDFS/OWL Constructs and User-Defined Rules in Oracle", Proceedings of the 24th International Conference on Data Engineering (ICDE 2008), Cancún, México, 7-12 April 2008, pp. 1239-1248.

[20] J. Lu et al., "SOR : a practical system for ontology storage, reasoning and search", Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB 2007), University of Vienna, Austria, 23-27 September 2007, pp. 1402-1405.

[21] IBM, "Developing RDF Applications for IBM Data Servers", January 2013. <ftp://ftp.software.ibm.com/ps/products/db2/info/vr101/pdf/en_US/DB2DevRDFdb2rdfe1011.pdf> [retrieved: June, 2015]

[22] A. Zekri, Z. Brahmia, F. Grandi, and R. Bouaziz, "τOWL: A Framework for Managing Temporal Semantic Web Documents Supporting Schema Versioning," International Journal On Advances in Software, in press.

[23] W3C, RDF/XML Syntax Specification (Revised), W3C Recommendation, 10 February 2004. <http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/> [retrieved: June, 2015]

[24] The Friend of a Friend (FOAF) project. <http://www.foaf-project.org/> [retrieved: June, 2015]

[25] W3C, Resource Description Framework (RDF), Semantic Web Standard. <http://www.w3.org/RDF/> [retrieved: June, 2015]

[26] M. Horridge and S. Bechhofer, "The OWL API: A Java API for OWL Ontologies", Semantic Web, vol. 2, February 2011, pp. 11-21.

[27] A. Zekri, Z. Brahmia, F. Grandi, and R. Bouaziz, "Temporal Schema Versioning in τOWL," Proceedings of the 2nd International Conference on Knowledge Management, Information and Knowledge Systems (KMIKS 2015), Hammamet, Tunisia, 16-18 April 2015, pp. 81-92.

 ISBN: 978-1-61208-420-6