τOWL: A Framework for Managing Temporal Semantic Web Documents

Abir Zekri
University of Sfax
Sfax, Tunisia
abir.zekri@fsegs.rnu.tn

Zouhaier Brahmia
University of Sfax
Sfax, Tunisia
zouhaier.brahmia@fsegs.rnu.tn

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

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

***Abstract*—The World Wide Web Consortium (W3C) OWL 2 Web Ontology Language (OWL 2) recommendation is an ontology language for the Semantic Web. It allows defining both schema (i.e., entities, axioms, and expressions) and instances (i.e., individuals) of ontologies. OWL 2 ontologies are stored as Semantic Web documents. However, OWL 2 lacks explicit support for time-varying schema or for time-varying instances. Hence, knowledge engineers or maintainers of semantics-based Web resources have to use ad hoc techniques in order to specify OWL 2 schema for time-varying instances. In this paper, for a disciplined and systematic approach to the temporal management of Semantic Web documents, we propose the adoption of a framework called Temporal OWL 2 (τOWL), which is inspired by the τXSchema framework defined for XML data. In a way similar to what happens in τXSchema, τOWL allows creating a temporal OWL 2 ontology from a conventional (i.e., non-temporal) OWL 2 ontology and a set of logical and physical annotations. Logical annotations identify which elements of a Semantic Web document can vary over time; physical annotations specify how the time-varying aspects are represented in the document. By using annotations to integrate temporal aspects in the traditional Semantic Web, our framework (i) guarantees logical and physical data independence for temporal schemas 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.**

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

I. INTRODUCTION

Time is an omnipresent dimension in both classical and modern applications [1]; it is used to timestamp data values to keep track of changes in the real world and model their history. Hence, studying time has been, and continues to be, one of the main research interests in different scientific fields, such as databases and knowledge representation.

Since the second half of the 1980s, a great deal of work has been done in the field of temporal databases [2][3][4]. Several data models and query languages have been proposed for the management of time-varying data. Temporal databases usually adopt one or two time dimensions to timestamp data: (a) transaction-time, which indicates when an event is recorded in the database, and (b) valid-time, which represents the time when an event occurred, occurs or is expected to occur in the real world.

On the other hand, the World Wide Web (WWW or Web) [5] was shifted from the semi-structured internet to a more structured Web called the Semantic Web [6][7]. The new generation of Web aims to provide languages and tools that specify explicit semantics for data and enable knowledge sharing among knowledge-based applications. In this vision, ontologies [8] are used for defining and relating concepts that describe Web resources, in a formal way. The new emerging standard for describing ontologies, which has been recommended by the W3C since 2009, is OWL 2 [9][10][11]. It allows defining both schema (in terms of entities, axioms, and expressions) and instances (i.e., individuals) of ontologies; OWL 2 ontologies are stored as Semantic Web documents.

Due to the dynamic nature of the Web, ontologies that are used on the Web (like other Web application components such as Web databases, Web pages and Web scripts) 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 [12], existing Semantic Web standards and 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 lacks explicit support for time-varying ontologies, at both schema and instance levels. Thus, knowledge engineers or maintainers 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. In the rest of the paper, we define as Knowledge Base Administrator (KBA) a knowledge engineer or, more in general, the person in charge of the maintenance of semantics-based Web resources.

According to what precedes, we think that if we would like to handle ontology evolution over time in an efficient manner and to allow historical queries to be executed on time-varying ontologies, a built-in temporal ontology

 ISBN: 978-1-61208-355-1

management system is needed. For that purpose, we propose in this paper 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 to the eXtensible Markup Language (XML) with Temporal XML Schema (τXSchema) [13][14][15]. τXSchema is a 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 [16]. Moreover, in our previous work [17][18][19], with the aim of completing the framework, we augmented τXSchema by defining necessary schema change operations acting on conventional schema, temporal schema, and logical and physical annotations (extensions which we plan to apply to τOWL too).

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 (i.e., non temporal) Semantic Web, our framework (i) guarantees logical and physical data independence [20] for temporal schemas 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.

The remainder of the paper is organized as follows. Section II motivates the need for an efficient management of time-varying Semantic Web documents. Section III describes the τOWL framework that we propose for extending the Semantic Web to temporal aspects: the architecture of τOWL is presented and details on all its components and support tools are given. Section IV discusses related work. Section V provides a summary of the paper and some remarks about our future work.

II. MOTIVATION

In this section, we present a motivating example that shows the limitation of the OWL 2 language for explicitly supporting time-varying instances. Then, we state the desiderata for an OWL 2 extension which could accommodate time-varying instances in a disciplined and systematic way.

A. Motivating Example

The Friend of a Friend (FOAF) project [21] is creating a Web of machine-readable pages describing people, the links between them and the things they create and do.

Suppose that the Web site “Web-S1” publishes the FOAF definition for his user “Nouredine”. A fragment of the FOAF Resource Description Framework (RDF) document of “Nouredine” is presented in Fig. 1. It describes, according to the FOAF ontology, the personal information of “Nouredine” (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 limit to describe user’s information concerning the account on the online Web site “Facebook”.

Assume that information about the user “Nouredine” of the Web site “Web-S1” was added on 2014-01-15. On 2014-02-08, Nouredine modified his nickname from “Nor” to “Nouri” and his account name of Facebook from “Nor_Tunsi” to “Nouri_Tunsi”. Thus, the corresponding fragment of the Nouredine FOAF RDF document was revised to that shown in Fig. 2.

```
…
<foaf:Person rdf:ID="#Person1">
  <foaf:name>Nouredine Tounsi</foaf:name>
  <foaf:nick>Nor</foaf:nick>
  <foaf:holdsAccount>
    <foaf:OnlineAccount
          rdf:about="https://www.facebook.com/
          Nouredine.Tounsi">
      <foaf:accountName>Nor_Tunsi
      </foaf:accountName>
    </foaf:OnlineAccount>
  </foaf:holdsAccount>
</foaf:Person>
…
```

Figure 1. A fragment of Nouredine FOAF RDF document on 2014-01-15.

```
…
<foaf:Person rdf:ID="#Person1">
  <foaf:name>Nouredine Tounsi</foaf:name>
  <foaf:nick>Nouri</foaf:nick>
  <foaf:holdsAccount>
    <foaf:OnlineAccount
          rdf:about="https://www.facebook.com/
          Nouredine.Tounsi">
      <foaf:accountName>Nouri_Tunsi
      </foaf:accountName>
    </foaf:OnlineAccount>
  </foaf:holdsAccount>
</foaf:Person>
...
```

Figure 2. A fragment of Nouredine FOAF RDF document on 2014-02-08.

In many Semantic Web-based applications, the history of ontology changes is a fundamental requirement, since such a history allows recovering past ontology versions, tracking changes over time, and evaluating temporal queries [22]. A τOWL time-varying Semantic Web document records the evolution of a Semantic Web document over time by storing all versions of the document in a way similar to that originally proposed for τXSchema [13].

Suppose that the webmaster of the Web site “Web-S1” would like to keep track of the changes performed on our FOAF RDF information by storing both versions of Fig. 1 and of Fig. 2 in a single (temporal) RDF document. As a result, Fig. 3 shows a fragment of a time-varying Semantic Web document that captures the history of the specified information of “Nouredine”.

```
…
<foaf:Person rdf:ID="#Person1">
  <foaf:name>Nouredine Tounsi</foaf:name>
  <versionedNick>
    <NickVersion>
```

```
      <nickValidityStartTime>2014-01-15
      </nickValidityStartTime>
      <nickValidityEndTime>2014-02-07
      </nickValidityEndTime>
      <foaf:nick>Nor</foaf:nick>
    </NickVersion>
    <NickVersion>
      <nickValidityStartTime>2014-02-08
      </nickValidityStartTime>
      <nickValidityEndTime>now
      </nickValidityEndTime>
      <foaf:nick>Nouri</foaf:nick>
    </NickVersion>
  </versionedNick>
  <foaf:holdsAccount>
    <foaf:OnlineAccount
          rdf:about="https://www.facebook.com/
          Nouredine.Tounsi">
      <versionedAccountName>
        <AccountNameVersion>
          <accountNameValidityStartTime>
               2014-01-15
          </accountNameValidityStartTime>
          <accountNameValidityEndTime>
               2014-02-07
          </accountNameValidityEndTime>
          <foaf:accountName>Nor_Tunsi
          </foaf:accountName>
        </AccountNameVersion>
        <AccountNameVersion>
          <accountNameValidityStartTime>
               2014-02-08
          </accountNameValidityStartTime>
          <accountNameValidityEndTime>
               now
          </accountNameValidityEndTime>
          <foaf:accountName>Nouri_Tunsi
          </foaf:accountName>
        </AccountNameVersion>
      </versionedAccountName>
    </foaf:OnlineAccount>
  </foaf:holdsAccount>
</foaf:Person>
...
```

Figure 3. A fragment of the time-varying Nouredine FOAF RDF document.

In this example, we use valid-time to capture the history of Nouredine information. In order to timestamp the entities which can evolve over time, we use the following optional tags: **nickValidityStartTime** and **nickValidityEndTime**, for recording **nick** name evolution, and **accountNameValidityStartTime** and **accountNameValidityEndTime**, for keeping the **accountName** history. These are optional Data Properties which can be added to a temporal entity. The domain of nickValidityEndTime or accountNameValidityEndTime includes the value "now" [23]; the entity that has now as the value of its validity end time property represents the current entity until some change occurs.

Assume that the extract of the FOAF ontology presented in Fig. 4 contains the conventional (i.e., non-temporal) schema [13] for the FOAF RDF document presented in both Fig. 1 and Fig. 2. The conventional schema is the schema for an individual version, which allows updating and querying individual versions.

```
<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 4. An RDF/XML extract from the OWL 2 FOAF ontology.

The problem is that the time-varying ontology document (see Fig. 3) does not conform to the conventional ontology schema (see Fig. 4). Thus, to resolve this problem, we need a different ontology schema that can describe the structure of the time-varying ontology document. This new schema should specify, for example, timestamps associated to entities, time dimensions involved, and how the entities vary over time.

B. *Desiderata*

There are several goals which can be fulfilled when augmenting the OWL 2 language to support time-varying instances. Our approach aims to satisfy the following requirements.

- Facilitating the management of time for KBAs.
- Supporting both valid time and transaction time.
- Supporting (temporal) versioning of OWL 2 instances.
- Keeping compatibility with existing OWL 2 W3C recommendations, standards, and editors, and not requiring any changes to these recommendations, standards, and tools.
- Supporting existing applications that are already using OWL 2 ontologies.
- Providing OWL 2 data independence so that changes at the logical level are isolated from those performed at the physical level, and vice versa.
- Accommodating a variety of physical representations for time-varying OWL 2 instances.

III. THE τOWL FRAMEWORK

This section presents our framework τOWL for handling temporal Semantic Web documents and provides an illustrative example of its use. It describes the architecture of τOWL and the tools used for managing both τOWL schema and τOWL instances. Since τOWL is a τXSchema-like framework, we were inspired by the τXSchema architecture and tools while defining the architecture and tools of τOWL.

 ISBN: 978-1-61208-355-1

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;
- use of temporal and physical annotations to specify temporal and physical aspects, respectively, at schema level.

Fig. 5 illustrates the architecture of τOWL. Notice that only the components which are shaded in the figure are specific to an individual time-varying OWL 2 document and need to be supplied by a KBA. The framework is based on the OWL 2 language [9], 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. Thus, we consider that the signature of an OWL 2 ontology O can be defined as follows: O = {E, A, Exp} such that:

i) E = {C, DP, OP, AP} represents the set of the entities with:
- C: Class, represents the set of concepts;
- DP: Data Property, represents the set of properties of the concepts;
- OP: Object Property, represents the set of the semantic relations between the concepts;
- AP: Annotation Property, represents the set of annotations on the entities and those on the axioms.

ii) A = {EAx, KAx} represents the set of axioms with:
- EAx: Entity Axioms, represents the axioms which concern the entities;
- KAx: Key Axioms, represents all the identifiers associated to the various classes.

iii) Exp = {CE, OPE, DPE} represents the set of the used expressions (an expression is a complex description which results from combinations of entities by using constructors such as enumeration, restriction of cardinality and restriction of properties) with:
- CE: Class Expressions, represents the set of combinations of concepts by using constructors;
- OPE: Object Property Expressions, represents the set of combinations of relations;
- DPE: Data Property Expressions, represents the set of combinations of properties.

The KBA starts by creating the *conventional schema* (box 6), 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 (i.e., non-temporal) OWL 2 instances (box 11). Any change to the conventional schema is propagated to its corresponding instances.

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.

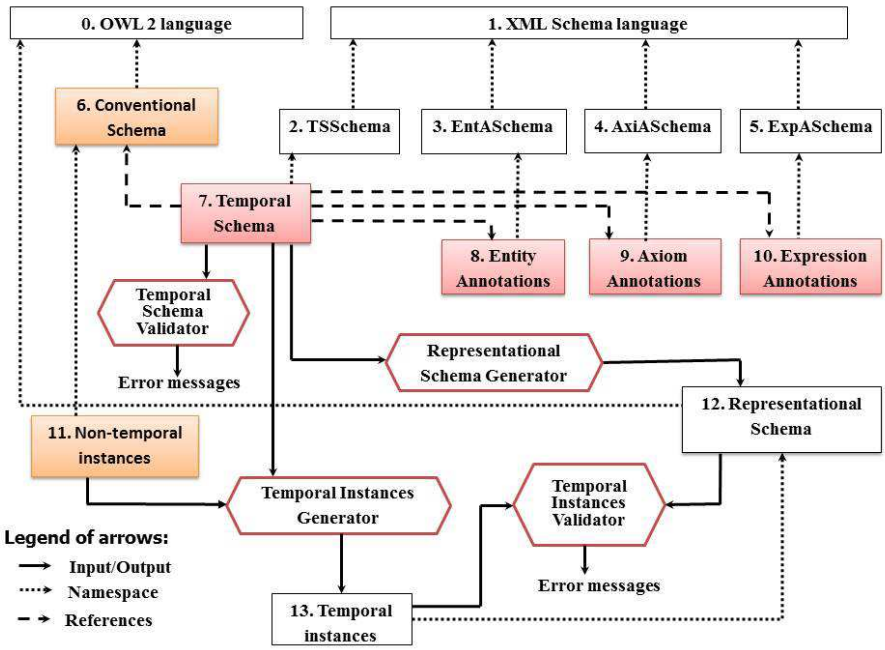

Figure 5. τOWL overall architecture.

Logical annotations [15] 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. If no logical annotations are provided, the default logical annotation is that anything can change. However, once the conventional schema is annotated, components that are not described as time-varying are static and, thus, they must have the same value across every instance document (box 11).

Physical annotations [15] 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. The location of timestamps is largely independent of which components vary over time. 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. Changing an aspect of even one timestamp can make a big difference in the representation. τOWL supplies a default set of physical annotations, which is to timestamp the root element with valid and transaction times. However, explicitly defining them can lead to more compact representations [15].

In order to improve conceptual clarity and also to enable a more efficient implementation, we adopt a "separation of concerns" principle in our approach: since the entities, the axioms and the expressions of an OWL 2 ontology evolve over time independently, we distinguish between three separate types of annotations to be defined and to be associated to a conventional schema: the *entity annotations* (box 8), the *axiom annotations* (box 9) and the *expression annotations* (box 10).

Entity annotations describe the logical and physical characteristics associated to the components of an OWL 2 ontology: classes, relations and properties. They indicate for example the temporal formats of these components which could be valid-time, transaction-time, bi-temporal or snapshot (by default). The schema for the logical and physical entity annotations is given by EntASchema (box 3). Axiom annotations and expression annotations describe the logical and physical aspects of axioms and expressions defined on classes or on properties. The schema for the logical and physical axiom annotations is given by AxiASchema (box 4) and the schema for the logical and physical expression annotations is given by ExpASchema (box 5).

Notice that AntASchema, AxiASchema, and ExpASchema, which all contain both logical and physical annotations, are XML Schemas [24]. The annotations associated to the same conventional schema can evolve independently. Any change to one of the three sets of annotations does not affect the two other sets.

Finally, the KBA creates the *temporal schema* (box 7) 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 2).

Notice that, whereas *TSSchema* (box 2), *AntASchema* (box 3), *AxiASchema* (box 4), and *ExpASchema* (box 5) have been developed by us, OWL 2 (box 0) and XML Schema (box 1) correspond to the standards endorsed by the W3C.

In a way similar to what happens in the τXSchema framework, the temporal schema document (box 7) is processed by the ***temporal schema validator*** tool in order to ensure that the logical and physical entity annotations, axiom annotations and expression annotations are (i) valid with respect to their corresponding schemas (i.e., *AntASchema*, *AxiASchema*, and *ExpASchema*, respectively), and (ii) consistent with the conventional schema. The temporal schema validator tool reports whether the temporal schema document is valid or invalid.

Once all the annotations are found to be consistent, the ***representational schema generator*** tool generates the *representational schema* (box 12) from the temporal schema (i.e., from the conventional schema and the logical and physical annotations); it is the result of transforming the conventional schema according to the requirements expressed through the different annotations. The representational schema becomes the schema for temporal instances (box 13). Temporal instances could be automatically created from the *non-temporal instances* (box 11) and the temporal schema (box 7), using the ***temporal instances generator*** tool (such an operation is called "squash" in the original τXSchema approach). Moreover, temporal instances are validated against the representational schema through the ***temporal instances validator*** tool which reports whether the temporal instances document (box 13) is valid or invalid.

Notice that the four mentioned tools (i.e., Temporal Schema Validator, Temporal Instances Validator, Representational Schema Generator, and Temporal Instances Generator) are under development. For example, the temporal instances validator tool is being implemented as a temporal extension of an existing conventional ontology instance validator.

Illustrative example. In order to show the functioning of the proposed approach, we provide in the following an example that shows how management of temporal ontology document versions is dealt with in our τOWL approach.

Let us resume the example of Sec. II.A. On 2014-01-15, the KBA creates a conventional ontology schema, named "PersonSchema_V1.owl" (as in Fig. 4), and a conventional ontology document, named "Persons_V1.rdf" (as in Fig. 1), which is valid with respect to this schema. Suppose that the

 ISBN: 978-1-61208-355-1

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 Fig. 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 6. The annotation document on 2014-01-15.

After that, the KBA creates the temporal ontology schema in Fig. 7, that ties "PersonSchema_V1.owl" and "PersonAnnotations_V1.xml" together; this temporal schema is saved in an XML file titled "PersonTemporalSchema.xml". Consequently, the Temporal Instances Generator tool uses the temporal ontology schema of Fig. 7 and the conventional ontology document in Fig. 1 to create a temporal document as in Fig. 8, 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 Fig. 9.

On 2014-02-08, the KBA updates the conventional ontology document "Persons_V1.rdf" as presented in Sec. II.A to produce a new conventional ontology document named "Persons_V2.rdf" (as in Fig. 2). 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 Fig. 10. The squashed version of the updated temporal document is provided in Fig. 11.

```
<?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 7. The temporal schema on 2014-01-15.

```
<?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 8. The temporal document on 2014-01-15.

```
<foaf:Person rdf:ID="#Person1">
  <foaf:name>Nouredine Tounsi</foaf:name>
  <nick_RepItem>
    <nick_Version>
      <timestamp_ValidExtent
              begin="2014-01-15" end="now" />
      <foaf:nick>Nor</foaf:nick>
    </nick_Version>
  </nick_RepItem>
  <foaf:holdsAccount>
    <foaf:OnlineAccount
          rdf:about="https://www.facebook.com/
          Nouredine.Tounsi">
      <accountName_RepItem>
        <accountName_Version>
          <timestamp_ValidExtent
                 begin="2014-01-15" end="now" />
          <foaf:accountName>Nor_Tunsi
          </foaf:accountName>
        </accountName_Version>
      </accountName_RepItem>
    </foaf:OnlineAccount>
  </foaf:holdsAccount>
</foaf:Person>
```

Figure 9. The squashed document correponding to the temporal document on 2014-01-15.

```
<?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 10. The temporal document on 2014-02-08.

```
<foaf:Person rdf:ID="#Person1">
  <foaf:name>Nouredine Tounsi</foaf:name>
  <nick_RepItem>
    <nick_Version>
      <timestamp_ValidExtent begin="2014-01-15"
                             end="2014-02-07" />
      <foaf:nick>Nor</foaf:nick>
    </nick_Version>
    <nick_Version>
      <timestamp_ValidExtent begin="2014-02-08"
                             end="now" />
      <foaf:nick>Nouri</foaf:nick>
    </nick_Version>
  </nick_RepItem>
  <foaf:holdsAccount>
    <foaf:OnlineAccount
          rdf:about="https://www.facebook.com/
          Nouredine.Tounsi">
```

```
      <accountName_RepItem>
        <accountName_Version>
          <timestamp_ValidExtent
                      begin="2014-01-15"
                      end="2014-02-07"/>
          <foaf:accountName>Nor_Tunsi
          </foaf:accountName>
        </accountName_Version>
        <accountName_Version>
          <timestamp_ValidExtent
                      begin="2014-02-08"
                      end="now" />
          <foaf:accountName>Nouri_Tunsi
          </foaf:accountName>
        </accountName_Version>
      </accountName_RepItem>
    </foaf:OnlineAccount>
  </foaf:holdsAccount>
</foaf:Person>
```

Figure 11. The squashed document correponding to the temporal document on 2014-02-08.

Obviously, each one of the squashed documents (see Fig. 9 and Fig. 11) should conform to a particular schema, i.e., the representational schema, which is generated from the temporal schema shown in Fig. 7.

IV. Related Work Discussion

OWL-Time (formerly DAML-Time) [25] is a temporal ontology that has been developed for describing the temporal content of Web pages and the temporal properties of Web services. Excepting language constructs for representing time in ontologies, mechanisms for representing evolution of concepts (e.g., events) over time are absent. Furthermore, temporal relations cannot be expressed directly in OWL, since they are ternary (i.e., properties of objects that change in time involve also a temporal value in addition to the object and the subject); representing such temporal relations in OWL requires appropriate methods (e.g., 4D-fluents [26]). Our approach allows KBA representing (i) evolution of concepts over time, and (ii) temporal relations.

In [27], the authors present the annotation features of OWL 2 by showing that this latter allows for annotations on ontologies, entities, anonymous individuals, axioms (e.g., giving information about who asserted an axiom or when), and annotations themselves. In our work, we took another direction from using OWL 2 annotation features because we rather wanted to exploit the power of the τXSchema approach (e.g. including the exploitation of a τXSchema-like underlying infrastructure).

Time dimension(s) are explicitly added to Semantic Web languages and formalisms (e.g., RDF, OWL, and SPARQL Protocol and RDF Query Language (SPARQL)) in order to represent time in semantic annotations, to build temporal ontologies and to support temporal querying and reasoning. An annotated bibliography of previous work in this area is presented in [12], and a survey on the models and query languages for temporally annotated RDF is provided in [37]. In particular, in the literature, there are various contributions that propose to represent temporal data in the Semantic Web.

Gutiérrez et al. [28] presented a comprehensive framework to incorporate temporal reasoning into RDF, yielding temporal RDF graphs. They define a syntactic notion of temporal RDF graphs. A powerful system, called CHRONOS, for reasoning over temporal information in OWL ontologies is presented in [38]. Since qualitative representations are very common in natural language expressions such as in free text or speech and can be proven to be valuable in the Semantic Web, the authors choose to represent both qualitative temporal (i.e., information whose temporal extents are unknown such as “before”, “after” for temporal relations) and quantitative information (i.e., where temporal information is defined precisely, e.g., using dates). The CHRONOS reasoner can be applied to temporal relations in order to infer implied relations and to detect inconsistencies while retaining soundness, completeness and tractability over the supported relations set. As opposed to Gutiérrez et al. [28] and Anagnostopoulos et al. [38], in our present approach, we are not interested in temporal reasoning (and, thus, in spatio-temporal reasoning).

A model of a multi-temporal RDF Schema (RDFS) database is proposed in [29] where the author considered that this database is a set of RDF triples timestamped along the valid and/or transaction time axes. To enable querying such a database, an extension of SPARQL language [30], called T-SPARQL, has been defined in [22]. The paper [31] proposes a logic-based approach to introduce valid-time into RDFS and OWL 2 languages. An extension of SPARQL that can be used to query temporal RDF(S) and OWL 2 is also presented. Moreover, the author describes a general query evaluation algorithm that can be used with all entailment relations used in the Semantic Web. Finally, he presents two optimizations of the algorithm that are applicable to entailment relations characterized by a set of deterministic rules, such RDF(S) and OWL 2 RL/RDF Entailment. In [32], the authors introduce “The Valid Ontology” approach as a temporal extension of OWL. Indeed, they propose to use a single temporal XML document to represent and store a multi-version ontology and use a temporal XML query processor to efficiently extract valid OWL ontologies from the XML document as temporal snapshots. The result is an efficient ontology temporal versioning solution, relying on standard XML technology. Two complementary and alternative proposals for modeling temporally changing information in OWL are proposed in [33]. They are based on the perdurantist theory and benefit from results coming from the discipline of Formal Ontology, in order to restrict the appropriate use of the proposed frameworks. In the first proposal, the authors combine the perdurantist worm view with the notion of individual concepts for formulating a conceptual structure that allows one to separate from the information that define all the individuals the information concerning those that can possibly change. In the second proposal, they extend the first proposal with the distinction between objects and moments and the notion of qua individuals, where a qua individual is the way an object participates in a certain relation. With regard to Grandi [29], Motik [31], Grandi et al. [32], and Zamborlini et al. [33], our approach does not deal with modeling of time inside the ontology. It just supports temporal versioning.

 ISBN: 978-1-61208-355-1

O'Connor et al. [34] present a methodology and a set of tools for representing and querying temporal information in OWL ontologies. Their approach uses a lightweight temporal model to encode the temporal dimension of data. It also uses the OWL-based Semantic Web Rule Language (SWRL) and the SWRL-based OWL query language (SQWRL) to reason with and query the temporal information represented using the proposed model. By now, our approach does not support temporally-aware semantic rules.

The authors of [35] propose a new language, called temporal OWL (tOWL), which is an extension of the Ontology Web Language Description Logics (OWL-DL) to the temporal aspect. It enables the representation of time and change in dynamic domains. Through a layered approach, they introduce three extensions: (i) Concrete Domains, which allow the representation of restrictions using concrete domain binary predicates, (ii) Temporal Representation, which introduces timepoints, relations between timepoints, intervals, and Allen's 13 interval relations [36] into the language, and (iii) TimeSlices/Fluents, which implement a perdurantist view on individuals and enable the representation of complex temporal aspects such as process state transitions. The main purpose of our approach is to support past ontology versions, to be accessed via time-slice queries. We think that supporting temporal ontology versions is very interesting for several purposes and in different areas. The problem of not having temporal versions is that, e.g., if we have now to investigate on someone having put some illegal material on Facebook last week, we want to be able to individuate the account details even if they have been changed thereafter.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed τOWL, a τXSchema-like framework, which allows creating a temporal OWL 2 ontology from a conventional OWL 2 ontology and a set of logical and physical annotations. Our framework ensures logical and physical data independence, since it (i) separates conventional schema, logical annotations, and physical annotations, and (ii) allows each one of these three components to be changed independently and safely. Furthermore, adoption of τOWL provides for 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. The extension of OWL 2 to temporal and versioning aspects is performed without having to depend on approval of proposed extensions by standardization committees (and on upgrade of existing tools conforming to standards to comply with approved extensions). In the next future, we intend to (i) study querying and updating instances of τOWL ontologies, and (ii) develop a prototype tool that shows the feasibility of our approach.

Our future work aims at extending τOWL to also support schema versioning [19][39] which is the most powerful technique for managing the history of schema changes, since (i) ontology schemata are also evolving over time to reflect changes in real-world applications [40], and (ii) keeping a fully fledged history of ontology changes, i.e. involving both the ontology instances and the ontology schema, is a required feature for many Semantic Web-based applications.

REFERENCES

[1] C. S. Jensen and R. T. Snodgrass, "Temporal Data Management," IEEE Transactions on Knowledge and Data Engineering, vol. 11, January/February 1999, pp. 36-44.

[2] O. Etzion, S. Jajodia, and S. Sripada (eds.), "Temporal Databases: Research and Practice," LNCS 1399, Springer-Verlag, 1998.

[3] C. S. Jensen and R. T. Snodgrass, "Temporal Database," in Liu L., Özsu M.T., (Eds.), Encyclopedia of Database Systems, Springer US, 2009, pp. 2957-2960.

[4] F. Grandi, "Temporal Databases," in M. Koshrow-Pour, (Ed.), Encyclopedia of Information Science and Technology (3rd Ed.), IGI Global, Hershey, in press.

[5] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret, "The World Wide Web," Communications of the ACM, vol. 37, August 1994, pp. 76-82.

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

[7] Semantic Web project. <http://www.w3.org/2001/sw/> [retrieved: July, 2014]

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

[9] W3C, OWL 2 Web Ontology Language – Primer (Second Edition), W3C Recommendation, 11 December 2012. <http://www.w3.org/TR/owl2-primer/> [retrieved: July, 2014]

[10] W3C, OWL 2 Web Ontology Language – Document Overview (Second Edition), W3C Recommendation, 11 December 2012. <http://www.w3.org/TR/owl2-overview/> [retrieved: July, 2014]

[11] W3C, OWL 2 Web Ontology Language – Profiles (Second Edition), W3C Recommendation, 11 December 2012. <http://www.w3.org/TR/owl2-profiles/> [retrieved: July, 2014]

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

[13] 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 the 9th International Conference on Extending Database Technology (EDBT 2004), Heraklion, Crete, Greece, 14-18 March 2004, pp. 348-365.

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

[15] F. Currim et al., "τXSchema: Support for Data- and Schema-Versioned XML Documents," TimeCenter Technical Report TR-91, 279 pages, September 2009. <http://timecenter.cs.aau.dk/TimeCenterPublications/TR-91.pdf> [retrieved: July, 2014]

[16] 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.

[17] Z. Brahmia, R. Bouaziz, F. Grandi, and B. Oliboni, "Schema Versioning in τXSchema-Based Multitemporal XML Repositories," Proceedings of the 5th IEEE International Conference on Research Challenges in Information Science (RCIS 2011), Guadeloupe - French West Indies, France, 19-21 May 2011, pp. 1-12.

[18] Z. Brahmia, F. Grandi, B. Oliboni, and R. Bouaziz, "Versioning of Conventional Schema in the τXSchema

 ISBN: 978-1-61208-355-1

Framework," Proceedings of the 8th International Conference on Signal Image Technology & Internet Systems (SITIS'2012), Sorrento – Naples, Italy, 25-29 November 2012, pp. 510-518.

[19] 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, in press, 2014. IGI Global.

[20] 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.

[21] The Friend of a Friend (FOAF) project. <http://www.foaf-project.org/> [retrieved: July, 2014]

[22] F. Grandi, "T-SPARQL: a TSQL2-like temporal query language for RDF," Proceedings of the 1st International Workshop on Querying Graph Structured Data (GraphQ 2010), Novi Sad, Serbia, 20 September 2010, pp. 21-30.

[23] J. Clifford, C. Dyreson, T. Isakowitz, C. S. Jensen, and R. T. Snodgrass, "On the Semantics of "Now" in Databases," ACM Transactions on Database Systems, vol. 22, June 1997, pp. 171–214.

[24] XML Schema Part 0: Primer Second Edition, W3C Recommendation, 28 October 2004. <http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/> [retrieved: July, 2014]

[25] W3C, Time Ontology in OWL, W3C Working Draft, 27 september 2006. < http://www.w3.org/TR/owl-time/ > [retrieved: July, 2014]

[26] C. A. Welty and R. Fikes, "A Reusable Ontology for Fluents in OWL," Proceedings of the 4th International Conference on Formal Ontology in Information Systems (FOIS 2006), Baltimore, Maryland, USA, 9-11 November 2006, pp. 226-236.

[27] W3C, OWL 2 Web Ontology Language – New Features and Rationale (Second Edition), W3C Recommendation, 11 December 2012. <http://www.w3.org/TR/owl2-new-features/> [retrieved: July, 2014]

[28] 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.

[29] F. Grandi, "Multi-temporal RDF ontology versioning," Proceedings of the 3rd International Workshop on Ontology Dynamics (IWOD 2009), Washington DC, USA, 26 October 2009. CEUR Workshop Proceedings (CEUR-WS.org), Vol-519. <http://ceur-ws.org/Vol-519/grandi.pdf> [retrieved: July, 2014]

[30] W3C, SPARQL Query Language for RDF, W3C Recommendation, 15 January 2008, <http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/> [retrieved: July, 2014]

[31] B. Motik, "Representing and Querying Validity Time in RDF and OWL: A Logic-based Approach," Proceedings of the 9th International Semantic Web Conference (ISWC 2010), Shanghai, China, 7-11 November 2010, pp. 550-565.

[32] 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.

[33] V. Zamborlini and G. Guizzardi, "On the representation of temporally changing information in OWL," Workshops Proceedings of the 14th IEEE International Enterprise Distributed Object Computing Conference (EDOCW 2010), Vitória, Brazil, 25-29 October 2010, pp. 283-292.

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

[35] 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.

[36] J. F. Allen, "Maintaining Knowledge About Temporal Intervals," Communications of the ACM, vol. 26, November 1983, pp. 832-843.

[37] A. Analyti and I. Pachoulakis, "A survey on models and query languages for temporally annotated RDF," International Journal of Advanced Computer Science and Applications, vol. 3, September 2012, pp. 28-35.

[38] 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.

[39] J. F. Roddick, "Schema Versioning," in Liu L., Özsu M.T., (Eds.), Encyclopedia of Database Systems, Springer US, 2009, pp. 2499-2502.

[40] D. Rogozan and G. Paquette, "Managing ontology changes on the semantic web," Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2005), Compiegne, France, 19-22 September 2005, pp. 430-433.

 ISBN: 978-1-61208-355-1