Tracing the Provenance of Object-Oriented Computations on RDF Data

Matthias Quasthoff, Christoph Meinel

Hasso Plattner Institute, University of Potsdam
{matthias.quasthoff, christoph.meinel}@hpi.uni-potsdam.de

Abstract. This paper presents a new method for tracing the provenance of RDF data within object-oriented computations. The proposed approach eliminates the burden of manually keeping track of data provenance from software developers. Using object triple mapping, the source data items used for generating result data can be identified efficiently. The integration into an existing object triple mapping framework shows the feasibility of the approach. The work presented will help developing compliant RDF-enabled applications using object-oriented programming and may support the efforts getting the Web of data mainstream.

1 Introduction

The meteoric rise of interoperable websites backed by massive amounts of user-generated data over the last decade clearly demonstrates people's interest in using integrated services over the World Wide Web. With the advent of user-generated content and so-called Social Web application programming interfaces it became also clear that some degree of decoupling between data on the Web and the services processing these data is desirable for a variety of reasons. For instance, users should not be forced to publish their data to some kind of "black box" Web services. Also, a shift from service control to user control on how user-generated data can be reused is desirable. The Linking Open Data initiative has shown that Semantic Web technologies such as the Resource Description Framework (RDF) and the SPARQL query language are the tools of choice for designing such interoperable Web applications. However, it is widely known that developing RDF-enabled applications usually requires some non-standard software engineering knowledge. This is still regarded one big hindering factor for getting Semantic Web technologies picked up by non-academic software engineers and a wider range of commercial products.

Besides the complexity introduced by using RDF technologies alone, using so-called linked data [1] from various data sources on the Web requires the consuming software to handle and process additional metadata about this data, mainly

- check data licenses and privacy policies attached to the data,
- assess provenance information about the data consumed and publish provenance information along with the data created,

– assess trust and quality of the data consumed, e. g., based on provenance information.

Such metadata processing is not necessary in traditional software projects building upon rather centralized data storage technologies. Hence it is considered something new and an extra-effort by software developers. We argue that besides making the development of RDF-enabled applications simpler, also higher-level operations like such metadata handling must be simplified for software developers. In this paper, we use Object Triple Mapping (OTM) [2–4], which has been proven to simplify the development of Semantic Web applications [5], and show how this approach does as well contribute to handling metadata in Semantic Web applications.

We show that Object Triple Mapping (OTM) is suitable to process and generate metadata required for distributed Web applications in object-oriented programs. OTM [4] lets developers focus on object-oriented concepts representing actual business logic and hides the most frequent RDF operations from the source code. In Section 3 we extend existing basic formalizations of OTM to be able to attach metadata to object-oriented data items. Afterwards in Section 4, we show how to pass such metadata through the object-oriented program's control flow. Section 4 also contains a detailed example of how provenance information will be collected during a specific object-oriented computation. The paper is concluded in Section 5.

2 Related work

This paper builds upon work from different fields of research on the Semantic Web. The Linking Open Data movement is driven by the observation that Semantic Web standards and technologies have matured to allow building real-world applications upon them, and that the publication of openly usable, real-world data on the Web might foster the development of such applications [6]. Inspite of the success of the initiative resulting in myriad widely usable datasets on the Web, we argue that the development of Semantic Web applications itself still needs to get simpler.

Lassila investigated the design of Semantic Web software, especially with regards to how to organize rather generic operations on RDF data such as reasoning and other operations more related to specific business logic [7]. Oren et al. address on a related problem: in their work on ActiveRDF they showed how to access RDF data in an object-oriented manner, hiding most basic access patterns to RDF from software developers [3]. In their work, they use the PathLog language [8] to explain the handling of object-oriented concepts. The idea of Object Triple Mapping (OTM) is partly inspired from similar work on accessing relational databases [9]. Other, more implementation-centric approaches to OTM include So(m)mer [2] and Elmo for the Java programming language, and Surf RDF for Python. An experimental evaluation showed that developing simple Semantic Web applications using OTM is much easier for the Semantic Web beginner and leads to better source code [5]. The actual semantics of

OTM and possible desirable extensions of the mapping are however not yet fully understood. This is still an open issue, especially to identify the most common patterns implemented in different types of Semantic Web software and separating the core concepts of OTM from optional, use-case-dependent extensions.

In this paper, we investigate how to deal with provenance information of RDF data in object-oriented programs. Provenance information helps consumers to assess data with regard to various issues: Is the data source trustworthy? Has the data been generated using high-quality algorithms? But also, are we allowed to use the data for our purposes? The latter question can, e. g., refer to data licenses or privacy policies. In this paper, we show how Hartig and Zhao's work on Web data provenance [10] can be integrated into OTM. We will mainly focus on the question what data items have been used to construct other data items, i. .e., tracking the input RDF data of an object-oriented computation and see the relation of this input to the output RDF data of the computation.

3 Object-oriented access to Web data formalized

To formally describe how to attach provenance information to the results of object-oriented computations on RDF data, a formal representation of the RDF and OO data models is required as well as a formal mapping between the two data models. In this section, the required formal concepts are presented. First, the RDF data model has an established formal notation building upon the following concepts [11].

Definition 1 (RDF data model) *Let U be the set of* URI references*, B an infinite set of* blank nodes*, and L the set of* literals*.*

- $V := U \cup B \cup L$ *is the set of RDF* nodes*,*
- $R := (U \cup B) \times U \times V$ *is the set of all* triples *or* statements*, that is, arcs connecting two nodes being labelled with a URI,*
- *any $G \subseteq R$ is an* RDF graph*.*

For their work on OTM, Oren et al. use PathLog [8] to describe object-oriented access to data [3]. PathLog itself differentiates between scalar and set-valued class members. For the scope of this paper however, dealing with set-valued class members will be sufficient. Also, we will use a simplified version of the *semantic structure* explained in [8].

Definition 2 (OO data model) *Let $\mathcal{N}$ be a set of names. The set of PathLog references $\mathcal{R}_{\mathcal{N}}$ is defined inductively as follows.*

- $n \in \mathcal{N}$ *is a reference, also called a* simple reference*.*
- *for references t_0, t_1 and a simple reference s*
  - $t_0.s$ *is a reference, called a* path
  - $t_0 : s$ *and* $t_0[s \rightarrow t_1]$ *are references, called* molecules*.*

Furthermore, a semantic structure is a triple $(\mathcal{N}, O, I)$ such that

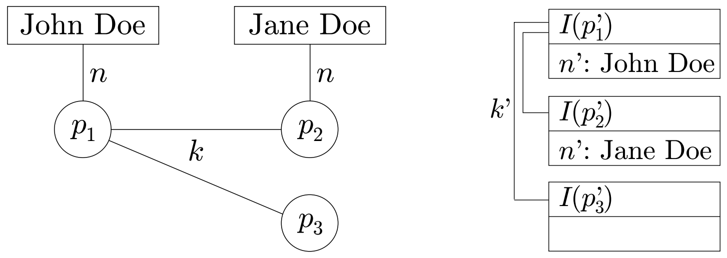

Fig. 1. Representing social information in RDF (left) and OOP (right).

- *O is a set of objects and*
- *The* interpretation *$I : \mathcal{R}_\mathcal{N} \to 2^O$ relates references to objects.*

In the following, we will show how RDF and PathLog can be used to express information in the respective data model.

Example 1 (comparison of RDF and OO data model) *Let p_1, p_2, $p_3 \in U$ be URI denoting three people, $n \in U$ be the URI* `foaf:name` *and $k \in U$ be the URI* `foaf:knows`[1]*. An RDF graph describing p_1 and p_2 might look the following (Fig. 1, left).*

$$G := \Big\{ \langle p_1, n, \text{“John Doe”} \rangle,\ \langle p_1, k, p_2 \rangle,\ \langle p_1, k, p_3 \rangle,\ \langle p_2, n, \text{“Jane Doe”} \rangle \Big\}$$

Let furthermore p_1', p_2', $p_3' \in \mathcal{N}$ be object names denoting three people and n', $k' \in \mathcal{N}$ fields labelled `name` *and* `knownPeople`*. The OO representation of G (Fig. 1, right) requires a semantic structure $(\mathcal{N}, O, I)$ such that*

$$\begin{aligned} I(p_1'.n') &= \{\text{“John Doe”}\} \\ I(p_2'.n') &= \{\text{“Jane Doe”}\} \\ I(p_1'.k') &= I(p_2') \cup I(p_3') \end{aligned}$$

The mapping of RDF data to OOP concepts as shown in Example 1 and vice versa has been formalized [5]. This formalization can be adopted to the concepts of PathLog as follows.

Definition 3 (Object triple mapping, OTM) *An* object triple mapping *for an RDF graph $G \subseteq (U \cup B) \times U \times V$ is a tuple $(\mathcal{N}, O, I, m_t, m_a)$, such that*

- *$(\mathcal{N}, O, I)$ is a semantic structure,*
- *the* vocabulary map *$m_t : F \to U$ maps a field names $F \subseteq \mathcal{N}$ to properties,*
- *the* instance map *$m_a : O \to U$ maps objects to resources,*
- *and the following holds for all $o \in O$, $n \in I^{-1}(\{s\})$, $f \in F$:*
  - *the mapping is complete:*
    $\forall u \in U : \langle m_a(o), m_t(f), u \rangle \in G \to \exists o' \in I(n.f) : m_a(o') = u$
    $\forall o' \in O : o' \in I(n.f) \to \langle m_a(o), m_t(f), m_a(o') \rangle \in G$

Matthias Quasthoff, Christoph Meinel:
"Tracing the Provenance of Object-Oriented Computations on RDF Data"
in Proceedings of the 2nd Workshop (in conjunction with 7th ESWC) on Trust and
Privacy on the Social and Semantic Web (SPOT 2010), 576, CEUR-WS, Heraklion, Greece, pp. paper 4, 5, 2010.

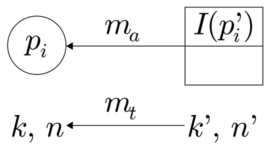

Fig. 2. Mapping OO concepts (right) on RDF (left).

- *the mapping is injective, i. e.* $m_{a\,|I(n.f)}$ *is injective.*

The idea of OTM is illustrated in Fig. 2 using the names from Example 1 and Definition 3. Besides such purely formal description of Object Triple Mapping, concrete implementations need to consider additional aspects such as object equivalence. Depending on the context, deciding on semantic equivalence can be hard. For the scope of this paper on metadata processing, considering syntactical equivalence based solely on the URI returned by m_a is sufficient. Also, concrete implementations need to map the semantics of RDF, such as lists or reification, of RDF Schema, such as class hierarchies, and OWL, such as constraints on classes and properties, to the semantics of their supported programming language. The general feasibility of such features is discussed in [12].

4 Tracing the provenance of object-oriented computations

4.1 Attaching metadata to RDF and OO data

RDF metadata will be attached to *data items* [10] like RDF graphs or triples. This applies for different kinds of metadata: Data licenses and policies specify whether a specific data item can be used in specific contexts or for specific purposes. Similarly, provenance metadata describe how a data item has been derived, and what other data items have been employed for such derivation. It is important to see that such kind of information must be attached to data items like RDF graphs or triples, not only to RDF resources. A similar concept is required to express metadata information about object-oriented concepts. In this section, we introduce the concept of *object-oriented data items.* The OO data items corresponding to RDF triples will be values together with the variables they are assigned to, or the values returned by a method together with the method called. Both cases can be represented by the PathLog reference used to obtain a certain object.

Definition 4 (OO data item) *Let* $(\mathcal{N}, O, I, m_t, m_a)$ *be an object triple mapping for some RDF graph* $G \subseteq (U \cup B) \times U \times V$*, and* $r \in \mathcal{R}_\mathcal{N}$ *a reference. Whenever we access an* $o \in I(r)$*, we say* o *has been obtained using the* OO data item $(r, I) \in \mathcal{R}_\mathcal{N} \times O^{\mathcal{R}_\mathcal{N}}$.

[1] The `foaf` prefix denotes the URI namespace http://xmlns.com/foaf/0.1/

References $r = n.f$ for names $n, f \in \mathcal{N}$ with n referring to a single object $I(n) = \{o\}$ are essential in object-oriented source code. Due to the nature of OTM, an object o' obtained using the data item $(n.f, I)$ has actually been obtained using the RDF triple

$$\langle m_a(o),\, m_t(f),\, m_a(o')\rangle$$

More complex references might be translated to other types of RDF data items, e. g., SPARQL query results. OO data items can now be used to process or update metadata attached to them. Such metadata can be obligations or restrictions of use due to privacy or license regulations, but also trust and quality assessments. Due to the mapping defined between the object-oriented and RDF data model, not only objects can be mapped to resources, but also OO data items to their corresponding RDF data items.

4.2 Aggregation of object-oriented provenance information

A significant part of tracing and generating provenance information is to relate the data items serving as input for a computation to its output data items. Such computation, or *data creation* [10], could, e. g., be performed by a method call in an OO program. However, from the object-oriented program flow within and around such methods further valuable information on data provenance can be aggregated.

Aggregation on data flow. If an object o has been obtained from a data item (r, I), a number of names appearing in r potentially identify objects $o'_1, \ldots, o'_\ell$ (a simple case is $r = n.f$, such that $I(n) = \{o'_1\}$). Naturally, all data items used to obtain $o'_1, \ldots, o'_\ell$ have also been used to eventually get hold of o.

Aggregation on object equality. A very common and important operation on objects o_1, o_2 is checking for equality. This can explicitly be triggered in source code by the software developer, but is also performed by higher-level operations provided by programming libraries, such as constructing the intersection of two sets of objects. If some action is taken because o_1 and o_2 have been detected being equal, all data items used to obtain o_1 and o_2, and the ones used to detect equality have actually been used to assign the value. Hence, for subsequent uses of o_1 and o_2, this aggregation of source data items should be considered. In implementations, this can be realized by adding side effects to equality checking.

4.3 Example: tracing provenance information of social network data

In this section, we present a comprehensive example of how to benefit from object-oriented provenance information on RDF data. The example illustrates that OO provenance information can be gathered during program execution without placing provenance-related artifacts in the OO source code. Given aquaintance information, we want to suggest new contacts for a person if the new contact is a mutual friend of at least to of the person's friends. The pseudo code for this computation is shown in Fig. 3. The relationships are described using the `foaf:knows` property k, and we introduce an additional RDF property s for suggested contacts. Consider the following RDF graph $G \subseteq (U \cup B) \times U \times V$ and person resources $p_1, \ldots, p_4 \in U$. If G contains

$$\langle p_1, k, p_2 \rangle, \quad \langle p_1, k, p_3 \rangle, \quad \langle p_2, k, p_4 \rangle, \quad \langle p_3, k, p_4 \rangle$$

then we want the suggestion $\langle p_1, s, p_4 \rangle$ to be created. Additionally, we want to be able to see the facts have been used to derive this statement, e. g., for later trust or quality assessments based on these provenance information.

To see what data items are used for the computation, let $(\mathcal{N}, O, I, m_t, m_a)$ be an object triple mapping and $o_1, \ldots, o_4, o'_4 \in O$ be objects representing $p_1, \ldots, p_4$. Going through the computation of `find_friends_for(`o_1`)` as described in Figure 3, the following OO provenance information is aggregated.

- In line 1, o_1 passed via `person`, i. e. $I(\texttt{person}) = o_1$.
- In line 2, the OTM interpretation returns $I(\texttt{person.friends}) = \{o_2, o_3\}$. The objects o_2, o_3 assigned to `friend1` will have the provenance information $(\texttt{person.friends}, I)$.
- In line 3, the loop-local interpretations return $I_1(\texttt{friend1.friends}) = \{o_4\}$, $I_2(\texttt{friend1.friends}) = \{o'_4\}$. The provenance information of o_4 will be $(\texttt{friend1.friends}, I_1)$ and for o'_4 it will be $(\texttt{friend1.friends}, I_2)$.
- In line 4, o_4 is not found equal to a friend of o_1.
- In line 6 in the first iteration, o_4 is not a friendship candidate, but is added to the candidates in line 9.
- In line 6 in the second iteration, o'_4 is found being equal to the friendship candidate o_4, hence o_4's and o'_4's provenance are both aggregated to $(\texttt{friend1.friends}, I_1)$, $(\texttt{friend1.friends}, I_2)$.
- In line 7 in the second iteration, o'_4 is suggested as a friend including this combined provenance information from line 6.

When the friend suggestion o'_4 to o_1 is mapped to the RDF statement $\langle p_1, s, p_4 \rangle$, this statement will eventually have been created using all OO data items and the corresponding RDF statements listed in Table 1. With the help of trust-

```
1   find_friends_for(person):
2       foreach friend1 in person.friends
3           foreach friend2 in friend1.friends
4               if person.friends contains friend2
5                   nothing to do, continue
6               else if candidates contains friend2
7                   add friend2 to suggestions
8               else
9                   add friend2 to candidates
```

Fig. 3. Object-oriented search for new contacts in a social network

OO data item	**RDF statement**	**Mapping from OO to RDF data**
$(\texttt{person.friends},\, I)$	$\langle p_1,\, k,\, p_2\rangle$ $\langle p_1,\, k,\, p_3\rangle$	$m_a(I(\texttt{person})) = \{p_1\}$ $m_t(\texttt{friends}) = k$ $m_a(I(\texttt{person.friends})) = \{p_2,\, p_3\}$
$(\texttt{friend1.friends},\, I_1)$	$\langle p_2,\, k,\, p_4\rangle$	$m_a(I_1(\texttt{friend1})) = \{p_2\}$ $m_t(\texttt{friends}) = k$ $m_a(I_1(\texttt{friend1.friends})) = \{p_4\}$
$(\texttt{friend1.friends},\, I_2)$	$\langle p_3,\, k,\, p_4\rangle$	$m_a(I_2(\texttt{friend1})) = \{p_3\}$ $m_t(\texttt{friends}) = k$ $m_a(I_2(\texttt{friend1.friends})) = \{p_4\}$

Table 1. Data items leading to the suggestion $\langle p_1,\, s,\, p_4\rangle$ in Fig. 3

related metadata potentially attached to these source triples, the result statement $\langle p_1,\, s,\, p_4\rangle$ can now be dealt with appropriately in further steps of the computation.

4.4 Implementation

The proposed approach for tracing the provenance of object-oriented computations has been integrated in our Java object triple mapping implementation OTMj [5]. In OTMj, an RDF resource u is mapped to a dynamic proxy object o implementing the interfaces corresponding to the known RDF types of u. This allowed for a straight-forward integration of the OO provenance model proposed: If o is accessed using a data item $(r,\, I)$, OTMj returns an object o' acting as a proxy for o. In addition to o, the proxy o' has the data item $(r,\, I)$ attached. OTMj can be obtained as open source software.[2]

Our implementation uses reification to link source and result triples using the concept of *data creations* as defined by Hartig and Zhao [10]. Given a triple $\langle s_1,\, p_1,\, o_1\rangle$ having been created using another triple $\langle s_2,\, p_2,\, o_2\rangle$, our implementation creates metadata as illustrated in Fig. 4[3].

5 Conclusion

In this paper we showed how metadata about RDF data can be processed and generated within object-oriented computations on these data using Object Triple Mapping (OTM). We extended existing formalisms to OTM and introduced the notion of object-oriented data items. Using these concepts we showed how to trace the provenance of RDF data items on their way through object-oriented computations and presented a detailed example and a ready-to-use implementation of the approach. Our next steps in research include a tighter integration with potential sources of provenance information such as SQUIN [13] or context-based

[2] http://projects.quasthoffs.de/otm-j
[3] The `prv` prefix denotes the URI namespace http://trdf.sourceforge.net/provenance/ns#

```
_:t1  a                 rdf:Statement ;
      rdf:subject       s1 ;
      rdf:predicate     p1 ;
      rdf:object        o1 ;
      prv:createdBy     _:c1 .
_:t2  a                 rdf:Statement ;
      rdf:subject       s2 ;
      rdf:predicate     p2 ;
      rdf:object        o2 .
_:c1  a                 prv:DataCreation;
      prv:usedData      _:t2 ;
      prv:performedBy   ... ;
      prv:performedAt   ... .
```

Fig. 4. N3 representation of metadata generated by OTMj during data creation.

reasoners [14], and with programming frameworks oriented towards generating complete applications instead of focusing on the data backend only, such as the Grails RDFa plugin[4].

This paper shows that the development of Semantic Web applications cannot only be simplified by focusing on established and widely understood abstractions like object-oriented programming, but also, functionality essential for a working Web of data can be integrated in these abstraction layers without adding further burden to software developers. Without these abstractions, metadata handling would have to be implemented per use-case, a task potentially being skipped due to time constraints or deferred for unlimited time by software engineers. We're looking forward to learning what will eventually make non-expert software developers use Semantic Web technologies and how our work will contribute to making this happen.

References

1. Berners-Lee, T.: Linked data. http://www.w3.org/DesignIssues/LinkedData.html (2006)
2. Story, H.: Java annotations and the semantic web. http://blogs.sun.com/bblfish/entry/java_annotations_the_semantic_web (2005)
3. Oren, E., Heitmann, B., Decker, S.: Activerdf: Embedding semantic web data into object-oriented languages. J. Web Sem. **6**(3) (2008) 191–202
4. Quasthoff, M., Meinel, C.: Design patterns for object triple mapping. In: Proc. of IEEE SCC 2009. (2009)
5. Quasthoff, M., Sack, H., Meinel, C.: How to simplify building semantic web applications. In: Proc. of the 5th International Workshop on Semantic Web Enabled Software Engineering, CEUR-WS.org (2009)
6. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: Proceedings of 6th International Semantic Web

[4] http://grails.org/plugin/rdfa

Conference, 2nd Asian Semantic Web Conference (ISWC+ASWC 2007). Springer (2008) 722–735
7. Lassila, O.: Programming semantic web applications: A synthesis of knowledge representation and semi-structured data. ISBN 978-951-22-8984-4 (2007)
8. Frohn, J., Lausen, G., Uphoff, H.: Access to objects by path expressions and rules. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: VLDB, Morgan Kaufmann (1994) 273–284
9. Fowler, M., Rice, D.: Patterns of Enterprise Application Architecture. Addison-Wesley (2003)
10. Hartig, O., Zhao, J.: Publishing and Consuming Provenance Metadata on the Web of Linked Data. In: Proceedings of the 3rd International Provenance and Annotation Workshop (IPAW). (2010)
11. Manola, F., Miller, E.: Rdf primer. w3c recommendation 10 february 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ (2004)
12. Quasthoff, M., Sack, H., Meinel, C.: Can software developers use linked data vocabulary? In: Proc. of I-Semantics '09. (2009)
13. Hartig, O., Bizer, C., Freytag, J.C.: Executing sparql queries over the web of linked data. In: Proc. of the 8th International Semantic Web Conference, Springer (2009)
14. Delbru, R., Polleres, A., Tummarello, G., Decker, S.: Context dependent reasoning for semantic documents in sindice. In: Proceedings of the 4th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2008), 7th International Semantic Web Conference, Kalrsruhe, Germany (10 2008)