SIOC in Action – Representing the Dynamics of Online Communities

Pierre Antoine Champin[1], Alexandre Passant[2]

[1]LIRIS, Université de Lyon, CNRS, UMR5205,
Université Claude Bernard Lyon 1, F-69622, Villeurbanne, France
`pchampin@liris.cnrs.fr`
[2]DERI, National University of Ireland, Galway
`alexandre.passant@deri.org`

Abstract. SIOC provides the Semantic Web with a vocabulary for representing the state of online communities at a given time. However, a number of web application put a strong emphasis on the *dynamics* of their components and users, as testified by the recent normalisation efforts on activity streams. This work proposes a module for the SIOC vocabulary designed to represent the dynamics of SIOC represented online communities, centred on the notion of action. We link this work to related vocabularies and technologies, both in use and emerging, inside and outside the field of Semantic Web technologies.

1 Introduction and Motivation

The linked data initiative [1] aims at providing on the Web machine-readable data, complementing the human-readable documents that constitutes the major part of the current Web. The goal is to allow computer-based agents to be much more efficient in assisting users and performing automated tasks on their behalf, thus realising the vision of the Semantic Web.

Although the amount of linked data has been growing quite impressively in the past few years[1], most of it is essentially *static* in nature. We do not mean that this data is not evolving, but that it describes the current state of their domain of interest. This contrasts with the importance of time and *dynamics* witnessed recently in popular Web applications.

Indeed, the advent of blogs in the late nineties has strongly shaped the Web by putting the temporal dimension forward, in primarily organising their content along a chronological line. Wikis also give a high importance to time, by keeping track of the modifications made on a page, and allowing readers to see the history of those changes. More recently, the Social Web has confirmed this interest in dynamics by providing users with a variety of tools for publishing information about their activity, and retrieving it from their social network. This is especially well illustrated by the initial tag line of the now leading micro-blogging service Twitter[2]: "What are you doing?".

[1] See `http://linkeddata.org/` for an up-to-date account of the linked-data cloud.
[2] `http://twitter.com/`

An important step in bridging the gap between the Semantic Web and the Social Web has been the SIOC ontology (Semantically Interconnected Online Communities). This vocabulary allows to expose various social Web applications (blogs, wikis, forums, etc.) on the Web of linked data. It is implemented in a number of popular softwares and frameworks (see `http://sioc-project.org/`).

In this paper, we introduce SIOC-actions, or `sioca` for short. It is a module for SIOC designed to represent how users of an online community are manipulating the various digital artifacts that constitute the application supporting that community. We believe that this kind of information can prove valuable in a large range of applications; for example, we have previously discussed the interest of *interaction traces* in fostering group reflexivity and group awareness in collaborative systems [2], providing a base for digital object memories [3], or supporting flexible automated reasoning [4].

The structure of the paper is the following: we first present briefly the SIOC ontology, before describing the `sioca` rationale and vocabulary. In section 4 we describe different scenarios making use of `sioca`, then in section 5 we compare it to related work. Finally, we conclude and provide some directions for further work.

2 The SIOC Vocabulary

The SIOC vocabulary [5] provides a generic vocabulary for representing Social Web applications. Figure 1 gives an overview of the different concepts and relations it defines. In this section, we briefly describe them as they constitute the background of our proposal.

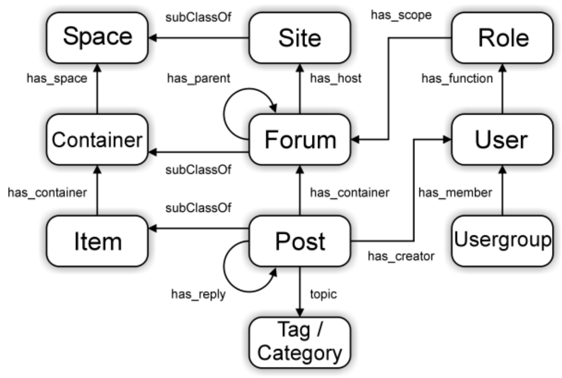

Fig. 1. Overview of the SIOC vocabulary

As can be seen in the left column of Figure 1, the atomic elements of the Web applications described by SIOC are called Items. They are grouped in Containers, that can themselves be contained in other Containers. Finally, every containers belongs to a Space. Those abstracts concepts are generic enough to represent a great variety of Web applications, but are best understood when instantiated into a concrete example, as the one given in the middle column: a Site may contain a number of Forums, some of them containing sub-forums, and every forum contains a set of Posts. Every Post (and actually every Item, Container or Space) can be associated with Tags or Categories representing their topic.

Finally, the right column of the figure shows how SIOC represents the Users creating Items. It is important to note that the class User does not intend to represent the physical people using the application, but rather the online account they are using. This distinction will have its importance in the definition of the sioca vocabulary.

3 The sioca Module

In this section we describe the sioca vocabulary. Note that in the following of the paper, we will use CURIE notations [6], and the following correspondance between CURIE prefix and namespace URI will be used.

dc:	`http://purl.org/dc/terms/`
event:	`http://purl.org/NET/c4dm/event.owl#`
foaf:	`http://xmlns.com/foaf/0.1/`
lode:	`http://linkedevents.org/ontology/`
sioc:	`http://rdfs.org/sioc/ns#`
time:	`http://www.w3.org/2006/time#`
$\emptyset$	`http://rdfs.org/sioc/actions#`

3.1 Actions and their Attributes

The central notion of the sioca module is the Action: a timestamped event involving one user and a number of digital artifacts. It is a subclass of the class event:Event described in the event ontology [7].

One point worth noting is that we consider actions to be instantaneous. Although we acknowledge the fact that some actions may actually take some time to be performed, we are only interested in the moment they are taking effect in the considered application. The rationale of this simplifying assumption is the following. First, all applications record the time when an action takes effect, while only some of them record how long it took to perform the action. Second, instantaneous actions are expected to be easier to manage in applications using our vocabulary.

We are not defining specific terms for expressing the time and agent of an action; we rather tap existing terms from related ontologies and vocabularies. Since they do not provide a single uniform way of representing that information, we discuss in the following how to reconcile them in the context of SIOC actions, which are illustrated in Figure 2.

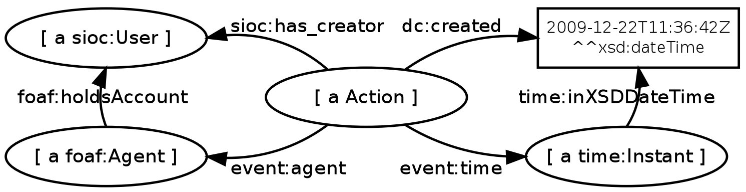

Fig. 2. Two representations of the actor and timestamp of an action

Per the event ontology, an event can be linked to its actors by the event:agent property, whose range is foaf:Agent. Though we are interested in this information, many application do not provide it, but only the *online account* used to perform the action. In the SIOC ontology, this notion is captured by the sioc:User class, which may be related to the corresponding agent with the foaf:holdsAccount property. We therefore require that each action be related to exactly one account through the sioc:has_creator property. We also define an axiom, using the property chain construct from OWL 2 [8], stating that the holder of the account is an event:agent of the action.

We have a similar discrepancy for representing the timestamp of an action (if at a structural rather than semantic level). In SIOC, the creation time of an item is usually represented by a literal and linked with property dc:created. On the other hand, the event ontology uses the event:time property to link to an *instance* representing either a time interval or an instant. The latter is overly complicated for our scenario, as we consider only instantaneous actions; furthermore, it can in principle be rebuilt from the information provided by dc:created.

However, this inference can not be represented using standard languages such as OWL 2 – though it could be represented with more expressive languages, such as SWRL [9]. It is therefore expected that implementers will use either one, the other or both representations, depending on the level of interoperability they aim to achieve with other vocabularies (see examples in Section 4.4).

3.2 The objects of Actions

The second important notion of our module is the one of DigitalArtifact, representing the objects manipulated through Actions. Its intent is to represent any component of the Web-based applications targeted by SIOC. It is therefore a superclass of most SIOC classes, such as sioc:Item and sioc:Space, but also sioc:User[3]. However, for the sake of openness, we do not want to restrict DigitalArtifact to those classes, as they may not cover other kinds of digital artifacts that may emerge in the future. It is not clear, for example, whether Facebook *applications* actually fit one of those classes; however, Facebook applications

[3] Recall that sioc:User do not represent persons, but rather their account in the application, which is indeed a digital artifact.

can obviously be *used*, an action that we should be able to represent with out vocabulary.

The digital artifacts manipulated by an action are called the *objects* of the action, as opposed to the *subject*, which is the actor performing the action. How an action relates to its objects is represented by a hierarchy of properties, rooted in the object property and described by figure 3. Note that an action may be related to several objects, with different subproperties of object for each of them.

```
object
 + creates → event:product
 |  + product
 |  \ byproduct
 |
 + uses ~ event:factor
 |  + source
 |  \ modifies
 |
 \ deletes
```

Fig. 3. The hierarchy of object properties

The first level of the hierarchy distinguishes three subproperties of object depending on the existence of the object before and after the action. Namely, creates implies that the object didn't exist before the action; uses, that it existed before and after the action; deletes, that it ceased to exist after the action. Of course, creates and deletes both imply that the action is a *cause* of this change in the existence of the object.

Property creates is a sub-property of event:product, which relates an event to "something produced during the event". Note that the properties are not equivalent because object (and hence creates) is restricted to digital artifacts, while event:product can point to any entity (a sound, an abstract situation, etc.). We also distinguish two sub-properties of creates: product and byproduct. The former implies that the creation of the object was the function or intent of the action. The latter, on the other hand, indicates that the creation of the object was only a secondary, or even unintended, effect of the action. For example, posting an image on a wiki will create the image resource (the product), but may also create an ancillary wiki-page describing the image (a byproduct). Depending on the context, this distinction may be hard to make or debatable; in those cases, implementers can still fall back to simply using creates.

Property uses relates an action to any digital artifact that is involved in an action without being created or deleted by it. It is related to event:factor, but has no obvious specialisation relation with that property: on the one hand, it is more specific as it is restricted to digital artifacts. On the other hand, it may be more general since event:factors are defined to be "passive factors" of an event. However, we do not exclude that some digital artifacts may have an active role in some actions (for example, bots on a IRC channel may react to a command issued by the user).

We also provide two useful sub-properties of uses. Property source is typically associated with a creates property; it means that the source object has served as a primary source of content for the created object. It allows to describe common use patterns on blogging ("reference", "traceback") and micro-blogging ("retweet") applications. Property modifies means that the resource was not only involved in the action, but also that it was significantly altered by it. What "significantly" really means depends of course on each application; for example, changing the license or visibility of an image on a photo-sharing application may not be considered as a significant change of the image (even if the underlying data structure has indeed been modified).

4 Using SIOC-actions

In this section we present how SIOC actions can be used to model different existing applications, and what benefit this brings to them.

4.1 Wikis

An important feature of some wikis such as Wikipedia[4] is that they keep track of all modifications made on their pages. While SIOC can be used to represent the current state of a wiki, sioca is therefore relevant for representing the history of each page. As a proof of concept, we developed a demo extracting the history of any Wikipedia page, and returning an RDF representation using the sioca vocabulary. This demo is available at `http://champin.net/wsgi/siocat/`.

This export of Wikipedia page histories considers two kinds of digital artifacts: pages and revisions. Pages are mutable objects, whose content changes over time, and which are identified by the usual Wikipedia URI (e.g. `http://en.wikipedia.org/wiki/Lyon`). Revisions are immutable objects, corresponding to one particular version of a page; they are identified by revision-specific URIs (e.g. `http://en.wikipedia.org/wiki/Lyon&oldid=332929880`). Unlike pages, revisions are never modified once created, they are merely deprecated by newer revisions.

Each action is therefore linked to the page (using the modifies property) and to the corresponding revision (using the creates property[5]). They are also linked to the timestamp of the action, as well as the Wikipedia account responsible for the modification.

[4] `http://en.wikipedia.org/`

[5] Whether the revision is a product or a byproduct depends on one's focus, so we do not commit to one or the other interpretation.

4.2 Developer Communities

Software developer communities have a history of focusing on actions. Indeed, version control systems (VCS), the central tools of those communities, are dedicated to keeping track of actions (usually named “commits” or “patches”) modifying a source tree. We demonstrate below how our vocabulary allows to capture the essential features of those actions, and hence to describe the content of VCS on the Web of linked data.

An important notion in modern VCS is that of *atomic commit*: related actions on multiple files should always be applied together. For example, the renaming of a function in a file is usually reflected in other files invoking that function; the modification must therefore be either applied on all files or not at all. By allowing an action to have several objects, with a different effect (creates, modifies, deletes) on each of them, our vocabulary is well adapted to represent atomic commits.

The notion of revision, as introduced above to represent wikis, is not always relevant in the context of VCS. It is so in centralised systems, such as Subversion[6], where each commit creates a new state of the source tree. Decentralised VCS such as Bazaar[7] or Git[8], on the other hand, allow each user to have their own version of the source tree depending on the patches they imported from various other users. There is no shared identification of states of the source tree. Consistency is guaranteed by maintaining an acyclic graph of *dependencies* between patches, so that importing a patch will require other ones on which it builds.

Although we do not define a specific term for representing dependancies between actions, the openness of RDF makes it quite easy to provide such a term in a separate vocabulary. Furthermore, other relations between VCS actions and other resources (such as tickets in a bug tracking system) could as easily be added, even if those resources were not hosted by the same application.

4.3 Social Websites

A number of social web applications put a growing emphasis on temporal information and actions. From specialised applications like LastFM[9] tracking music listening habits, to general purpose statuses in Twitter or Facebook[10], a number of users’ actions are recorded and broadcast to their social network.

Since those actions take place in the digital environment of social Web applications, they invariably involve digital artifacts (contents, comments, statuses), if only to *use* them (such as a music file or a facebook application). As with developer communities, our vocabulary is a generic base that can be used as is

[6] `http://subversion.tigris.org/`
[7] `http://bazaar-vcs.org/`
[8] `http://git-scm.com/`
[9] `http://lastfm.com/`
[10] `http://facebook.com`

to provide a generic description of actions in social applications, or extended to provide more detail (see Section 5.3).

4.4 Querying SIOC-actions

The first benefit of a common language to represent actions is obviously the increased interoperability of the tools allowing to cope with that language, such as queries, presentation templates, etc. Using an RDF-based language allows, in turn, to use standard and generic technologies to build those tools, like the SPARQL query language [10], or Fresnel [11] or Tal4Rdf [12] for presentation.

Another benefit brought by RDF, which we already pointed out regarding VCS, is its inherent openness. It first allows to mix different vocabularies in order to capture several aspects of the same resource. Second, it allows to link resources from different applications in order to elicit relations that are at best implicit in each separate application [1]. Not only can information be exported outside the boundaries of information silos, but links across sources augments the value of each source by virtue of the network effect.

Figures 4 and 5 provide some example queries that could be used, provided a repository using the sioca vocabulary. Query (a) retrieves recent actions involving a given person, regardless of the the various applications this person may have used – as long as we can relate his different online accounts to the URI identifying him as a person.

Query (b) retrieves all the actions performed on a given blog post. Those actions may include the creation and updates of this post by its author, comments made by readers on the same blog. But it may as well include comments made in other blogs and quoting that post, the creation of shared bookmarks pointing to it, etc. The only limit is the number of action sources available to the query engine.

Note that those queries assume a fixed representation of time: using the event ontology for query (a), or the dublin core vocabulary as suggested by SIOC for query (b). Query (c) gives another example retrieving all actions performed on Christmas day, regardless of their representation of time.

Those examples only scratch the surface of how SIOC actions can be used to answer complex queries on a collection of actions spanning across application. One could query for all persons having modified two given wiki pages, as illustrated by query (d). Linking to other data from the linked data cloud, one could look for people having written a scientific paper related to a given topic, and having also altered the wikipedia page of this topic, as shown in query (e).

Another example in the field of VCS could be: finding all the persons having solved a bug that they had raised themselves. In the field of social networks, one could ask what kind of music has been recently listened by their friends, looking for "friendship" relations in different social applications. Finally, one could imagine to ask what kind of music is most frequently listened by people fixing bugs in a particular software...

```
DESCRIBE ?action
WHERE {
  <http://champin.net/foaf.rdf#pa> foaf:holdsAccount ?user .

  ?action event:time ?instant ;
          sioc:has_creator ?user .

  ?instant time:inXSDDateTime ?when .
  FILTER (?when > "2009-11-30"^^xsd:date)
}
```

(a) Querying all recent actions by Pierre-Antoine Champin.

```
SELECT ?who ?what ?when
WHERE {
  ?action dc:created ?when ;
          sioc:has_creator ?user ;
          ?what
<http://apassant.net/blog/2009/11/26/decoding-short-urls> .

  ?what rdfs:subpropertyOf :object .

  ?agent foaf:holdsAccount ?user ;
         foaf:name ?who .
}
ORDER BY ?when
```

(b) Querying all actions on a blog post.

```
DESCRIBE ?action
WHERE {
  OPTIONAL {
      ?action dc:created ?when .
  }
  OPTIONAL {
      ?action event:time ?instant.
      ?instant time:inXSDDateTime ?when .
  }
  FILTER (
      bound(?when)
   && ?when > "2009-12-24"^^xsd:date
   && ?when < "2009-12-26"^^xsd:date
  )
}
```

(c) Querying all actions performed on Christmas day.

Fig. 4. Simple SPARQL queries over SIOC actions

```
SELECT ?who
WHERE {
    ?action1 :modifies <http://fr.wikipedia.org/wiki/Lyon> ;
             sioc:has_creator :user1 .
    ?action2 :modifies <http://en.wikipedia.org/wiki/Lyon> ;
             sioc:has_creator :user2 .
    ?who foaf:holdsAccount ?user1, ?user2 .
}
```

(d) Querying all persons having modified both the french and english versions of the Wikipedia page of Lyon.

```
SELECT ?who
WHERE {
    ?who foaf:holdsAccount ?user .

    ?action :modifies ?page ;
             sioc:has_creator :user .

    ?paper a swrc:InProceedings ;
           dc:creator ?who ;
           foaf:topic ?topic .

    ?topic foaf:page ?page .
}
```

(e) Quering all persons having written about a topic and modified that topic in Wikipedia.

Fig. 5. More SPARQL queries over SIOC actions

5 Related Work

5.1 Representation of Events

We have chosen to base our vocabulary on the event ontology [7]. This ontology is not by far the only one available to represent events in RDF, as pointed out by Shaw *et al.* [13]. The authors of that paper give a nice overview of different existing proposals, their different ontological commitments, and provide a pivot vocabulary: the LODE ontology.

The reasons why we kept the event ontology as a reference for our own vocabulary are the following. First, the comparison proposed by Shaw *et al.* confirmed that its ontological commitment and level of detail fitted our needs better than other proposals (except for LODE itself, see below). Second, this vocabulary is still widely used in the linked data community, and interoperability may be easier to reach with that vocabulary than with LODE.

Finally, all the terms, but one, that we are borrowing from the event ontology, are semantically *equivalent* to a term from the LODE ontology. Should LODE gain popularity over the event ontology, the migration to that new vocabulary would therefore be straightforward. The only exception is lode:involved, which has no direct correspondence to sioc:factor, and contrarily to the latter, can be considered as a super-property of object. We have therefore included this axiom in our ontology.

5.2 Limitations of SPARQL

The examples from the previous section raise a number of problems that are outside the scope of this paper, but have been to some extent addressed elsewhere.

First, it may seem from our example queries that they required all actions to be stored in a single repository, which is obviously unrealistic if we aim at targeting a great number of Web-scale applications. However, several proposals [14, 15] have been made to execute SPARQL queries over the whole Web of linked data by discovering relevant sources opportunistically.

Another problem with querying SIOC actions with SPARQL is that SPARQL is usually not considered as a good fit to querying temporal data. Indeed, built-in operators on temporal datatypes only allow for comparison; computing the duration between two actions is not feasible in standard SPARQL. Furthermore, even comparing events temporally can become quite complex, as illustrated in Figure 6.

Several proposals have been made to enhance SPARQL [16] or to provide a dedicated query language [17]. However, those efforts consider time not as a part of RDF data, but as an orthogonal dimension, where each RDF triple is only valid during a certain period of time. They are therefore not directly applicable to our scenarios.

```
?action1 dc:created ?when1 .
?action2 dc:created ?when2 .
?action3 dc:created ?when3 .
FILTER (?when1 < ?when2 && ?when2 < ?when3 )
```

Fig. 6. A SPARQL WHERE clause stating that action2 must happen between action1 and action3.

5.3 Activity Streams and Traces

Aside from the linked data and semantic Web communities, key players in social Web applications have also seen the benefits in fostering interoperability and exchanging information about actions and activities. This has resulted in a joint effort to define a common format for Activity Streams [18, 19]. The proposed format is based on the popular XML-based Atom format: each activity (equivalent to our notion of Action) is described by an Atom entry extended with specific elements and attributes.

This format being based on XML, we argue that it not as prone as sioca to integrate with the growing amount of RDF-based linked data. Although the underlying conceptual model seems[11] quite similar to ours, activities are restricted to involve exactly one object. As illustrated by the examples in section 4, we consider this restriction to be too strong.

On the other hand, Activity Streams define a number of interesting notions that are not currently present in our vocabulary. A *stream* is a collection of activities, providing general information about the activites it contains. In particular, a stream has a *subject*, which can be either a digital artifact or an agent, and is the common feature between all the activities of the stream. Activities may also have a *target*, e.g. the album to which a photo is posted. This, as well as an extensive list of *verbs* defined in [19], are good candidates for defining sub-properties of object.

6 Conclusion and Further Work

In this paper, we have proposed a vocabulary to represent the dynamics of online communities, by representing the actions performed by the users of those communities. This vocabulary complements the SIOC ontology, a popular vocabulary for representing the static features of online communities. We illustrated how actions can be modelled and queried in different kinds of online communities.

As discussed in section 4, further refinement of our vocabulary may be required by different applications. The Activity Streams [19] effort provides a base for such extensions in the domain of social applications. We are considering to

[11] No conceptual model is explicitly described; it can only be inferred from the proposed syntax.

propose an extension of both SIOC and sioca dedicated to developer communities. This will relate to other existing vocabularies, such as DOAP[12].

As Activity Streams are rapidly gaining momentum in the domain of social Web applications, we are also planning to provide translation services between both formats. As stated in Section 5.3, there is indeed a large overlap between them.

Finally, we will continue to explore innovative applications consuming actions in the form of interaction traces, as the ones proposed in [2–4]. From that perspective, not only would SIOC-actions add a new dimension to the Web of linked data, but their relations to existing linked data would open new perspectives to trace-based applications.

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data – The Story So Far. International Journal on Semantic Web & Information Systems **5**(3) (2009) 1–22
2. Clauzel, D., Sehaba, K., Prié, Y.: Modelling and visualising traces for reflexivity in synchronous collaborative systems. In: International Conference on Intelligent Networking and Collaborative Systems (INCoS 2009), IEEE Computer Society (November 2009) 16–23
3. Settouti, L.S., Prié, Y., Cram, D., Champin, P.A., Mille, A.: A trace-based framework for supporting digital object memories. In: 1st International Workshop on Digital Object Memories (DOMe'09) in the 5th International Conference on Intelligent Environments (IE 09). (July 2009)
4. Cordier, A., Mascret, B., Mille, A.: Extending Case-Based Reasoning with Traces. In: Grand Challenges for reasoning from experiences, Workshop at IJCAI'09. (July 2009)
5. Breslin, J.G., Harth, A., Bojars, U., Decker, S.: Towards semantically-interlinked online communities. In Gómez-Pérez, A., Euzenat, J., eds.: ESWC. Volume 3532 of Lecture Notes in Computer Science., Springer (2005) 500–514
6. Birbeck, M., McCarron, S.: CURIE syntax 1.0. W3C working draft, W3C (March 2007)
7. Raimond, Y., Abdallah, S.: The Event Ontology (October 2007) `http://motools.sf.net/event/event.html`, accessed on 22/12/2009.
8. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.: OWL 2 web ontology language primer. W3C recommendation, W3C (October 2009) `http://www.w3.org/TR/owl2-primer/`.
9. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: a semantic web rule language combining OWL and RuleML. W3C member submission, W3 (May 2004) `http://www.w3.org/Submission/SWRL/`.
10. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C recommendation, W3C (2008) `http://www.w3.org/TR/rdf-sparql-query/`.
11. Pietriga, E., Bizer, C., Karger, D., Lee, R.: Fresnel: A browser-independent presentation vocabulary for RDF. In: Lecture Notes in Computer Science. Volume 4273., Athens, GA, USA (November 2006) 158

[12] `https://trac.usefulinc.com/doap`

12. Champin, P.A.: T4R: Lightweight presentation for the Semantic Web. In Chris Bizer, Sören Auer, G.A.G., ed.: Scripting for the Semantic Web, workshop at ESWC 2009. (June 2009)
13. Shaw, R., Troncy, R., Hardman, L.: LODE: Linking open descriptions of events. In Gómez-Pérez, A., Yu, Y., Ding, Y., eds.: ASWC. Volume 5926 of Lecture Notes in Computer Science., Springer (2009) 153–167
14. Bojars, U., Passant, A., Giasson, F., Breslin, J.G.: An architecture to discover and query decentralized rdf data. In Auer, S., Bizer, C., Heath, T., Grimnes, G.A., eds.: SFSW. Volume 248 of CEUR Workshop Proceedings., CEUR-WS.org (2007)
15. Hartig, O., Bizer, C., Freytag, J.C.: Executing SPARQL queries over the web of linked data. In Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K., eds.: International Semantic Web Conference. Volume 5823 of Lecture Notes in Computer Science., Springer (2009) 293–309
16. Tappolet, J., Bernstein, A.: Applied temporal rdf: Efficient temporal querying of rdf data with sparql. In Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E.P.B., eds.: ESWC. Volume 5554 of Lecture Notes in Computer Science., Springer (2009) 308–322
17. Baratis, E., Petrakis, E.G.M., Batsakis, S., Maris, N., Papadakis, N.: Toql: Temporal ontology querying language. In Mamoulis, N., Seidl, T., Pedersen, T.B., Torp, K., Assent, I., eds.: SSTD. Volume 5644 of Lecture Notes in Computer Science., Springer (2009) 338–354
18. Atkins, M., Recordon, D., Messina, C., Keller, M., Steinberg, A., Dolin, R.: Atom activity extensions (Draft). Internet-Draft (September 2009) `http://martin.atkins.me.uk/specs/activitystreams/atomactivity`.
19. Atkins, M., Recordon, D., Messina, C., Keller, M., Steinberg, A., Dolin, R.: Atom activity base schema (draft). Internet-Draft (December 2009) `http://martin.atkins.me.uk/specs/activitystreams/activityschema`.