Structured Feedback

A Distributed Protocol for Feedback and Patches on the Web of Data

Natanael Arndt* Kurt Junghanns Roy Meissner

Philipp Frischmuth Norman Radtke Marvin Frommhold

Michael Martin

Agile Knowledge Engineering and Semantic Web (AKSW)
Institute of Computer Science
Leipzig University
Augustusplatz 10
04109 Leipzig, Germany
{arndt|kjunghanns|meissner|frischmuth|radtke|frommhold|martin}@informatik.uni-leipzig.de

ABSTRACT

The World Wide Web is an infrastructure to publish and retrieve information through web resources. It evolved from a static *Web 1.0* to a multimodal and interactive communication and information space which is used to collaboratively contribute and discuss web resources, which is better known as *Web 2.0*. The evolution into a Semantic Web (*Web 3.0*) proceeds. One of its remarkable advantages is the decentralized and interlinked data composition. Hence, in contrast to its data distribution, workflows and technologies for decentralized collaborative contribution are missing. In this paper we propose the *Structured Feedback* protocol as an interactive addition to the *Web of Data*. It offers support for users to contribute to the evolution of web resources, by providing structured data artifacts as patches for web resources, as well as simple plain text comments. Based on this approach it enables crowd-supported quality assessment and web data cleansing processes in an ad-hoc fashion most web users are familiar with.

CCS Concepts

• **Information systems** → **Web applications; Collaborative and social computing systems and tools;** *Web services; Resource Description Framework (RDF);*

Keywords

linked data, dssn, semantic pingback, pubsubhubub, feedback, distributed semantic services, quality, crowd sourcing

1. INTRODUCTION

The Internet and the World Wide Web (WWW) in particular provide an infrastructure for retrieving information through web resources. With the advent of discussion forums, blogs, short message services like Twitter and online social networks like Facebook, the web evolved towards the *Web 2.0*. With the Semantic Web resp. Web of Data, the web is now evolving towards a *Web 3.0* consisting of structured and linked data resources (*Linked Data*, cf. [3]). Even though *Web 2.0* services can be transformed to provide Linked Data [7], there is a need for services to collaboratively interact with meaningful data and enable (Linked) Data Services to be integrated into the existing social web stack. Currently the Distributed Semantic Social Network (DSSN, section 3.1 and [18]) is available as a framework resp. platform to build social services based on the Web of Data. In addition to providing static RDF datasets, which can be queried as Linked Data or via a SPARQL service, it enables active communication along edges of the Giant Global Graph[1] and a "follow"-function for resources.

Building on the existing stack, we propose an open protocol on the Web of Data, which is distributed across multiple services, to integrate a collaborative feedback mechanism on structured and unstructured web resources. The protocol integrates elements of the architecture for a Distributed Semantic Social Network (cf. [18]), in particular the Semantic Pingback protocol (cf. [17]) and can be extended by a publish-subscribe system using PubSubHubbub (cf. [8]).

Current mechanisms to provide feedback on web resources are, but are not limited to, leaving a comment through a form provided by a publisher of a webpage, publishing a comment along with a link to the web resource on an online social network or doing so on a personal weblog. The problems with this kind of commenting are: either it is hard for the publisher of the web resource to get an overview of the available feedback to his content, or the user who leaves the comment is limited to the possibilities provided by the publisher to express the feedback and is not able to select the location of the comment and maybe specify the audience on her own. In contrast to this situation, we are proposing a crossover approach, which serves the upsides of the cur-

*corresponding author

WWW2016 Workshop: Linked Data on the Web (LDOW2016)
April 12, 2016, Montréal, QC, Canada

[1] http://dig.csail.mit.edu/breadcrumbs/node/215

rent situation to both parties, commenters and publishers. It gives commenters the possibility to publish comment resources at their leisure in an arbitrary location on the web. It is integrated in the DSSN and has the possibility to be integrated in further online social networks. Publishers are actively informed about distributed comments on the web. In addition, it provides better possibilities to express the user's feedback by creating structured patches to be sent to the publisher, which in turn are easier to apply than a verbal description about a mistake.

Nevertheless, the social network is just a part of the WWW and the Linked Open Data Cloud [4, 15]. Nowadays the Web of Data is a big worldwide database of structured data in a varying quality, which is constantly growing by human as well as machine generated data. This global graph covers a huge amount of topics, disciplines and areas and is highly interlinked [4, 15]. But this graph is mainly static and there are no commonly established practices for the crowd to contribute to the data quality or content of the data sets in an ad-hoc way, as it is possible for the Wikipedia or in online social networks. With the ability of our approach to provide structured patches to web resources, crowd based quality assessment and web data cleansing processes become feasible.

The paper is structured as follows: general requirements to a distributed structured feedback protocol are formulated in section 2, followed by relevant preliminaries, such as the DSSN, Semantic Pingback, PubSubHubbub and our interpretation of Linked Data, in section 3. The proposed protocol is specified in detail in section 4. Further we are documenting our reference implementation in section 5, which is the basis for our demonstration and evaluation in section 6. We are discussing the state of the art and related work in section 7. Finally a conclusion and an outlook on future work is given in section 8.

Throughout the paper we are referring to several RDF terms using simplified QNames[2]. The prefix definitions are as follows:

```
@prefix fb-res:        <http://resource.feedback.aksw.org/> .
@prefix fb-feed:           <http://feed.feedback.aksw.org/> .
@prefix dbpedia:             <http://dbpedia.org/resource/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns\#> .
@prefix foaf:                  <http://xmlns.com/foaf/0.1/> .
@prefix sioc:                   <http://rdfs.org/sioc/ns\#> .
@prefix sioct:                <http://rdfs.org/sioc/types\#> .
@prefix pingback:           <http://purl.org/net/pingback/> .
@prefix cs:      <http://purl.org/vocab/changeset/schema\#> .
@prefix prov:                  <http://www.w3.org/ns/prov\#> .
@prefix xsd:           <http://www.w3.org/2001/XMLSchema\#> .
```

2. REQUIREMENTS

In the following we present the requirements for an open federated commenting system on the web. The requirements are targeted by our proposed architecture and protocol through different components.

Decentralized Storage of Comment Resources.

Comment Resources should be stored in any location on the web. Preferably the location should be publicly available and has to be identifiable and locatable by a URL. The location of a *Comment Resource* should be selected by its author.

Structured Feedback for Web Resources.

Besides the classic way of giving feedback for a *Web Resource* by adding a simple full text comment it should be possible to give a more precise structured comment to a resource. This enables users to directly fix the mistake and provide the resource publisher with a patch to apply to the *Web Resource*, instead of verbally describing the position of a mistake in a document. This behavior is comparable to the concept of pull-requests in software engineering e. g. on git-based collaboration platforms.

Comment Container for a Web Resource.

All comments related to a specific *Web Resource* have to be aggregated in a single place, which is referenceable from this *Web Resource*. This aggregated resource has to contain references to the original comments. The *Comment Container* can be an RDF resource or an RSS web feed (cf. [1]).

Active Updating of the Comment Container with new Comment Resources.

The *Comment Container* has to be updated if new *Comment Resources* for the respective *Web Resource* are created. This is needed to have an up-to-date resource available to the content publisher to get informed about new comments and be able to integrate structured patches into his data set.

Active Notification for Subscribers of Comments on Web Resources.

Web users which are interested in a specific *Web Resource* should be able to subscribe to a *Comment Container* related to the *Web Resource* of interest. This *Comment Container* should be kept up to date regarding new comments. Subscribers have to be actively informed of updates on *Comment Containers* they are subscribed to.

3. PRELIMINARIES

The proposed architecture and protocol is based on several existing technologies and protocols used on the Web of Data. From a general perspective it is based on HTTP. Specifically it is relying on and following the widely used Linked Data principles (cf. [3]), which are slightly extended to support our needs (cf. section 3.4). Since this proposal is partially located in the area of social networks it comprises the two main protocols from the Distributed Semantic Social Netwok (DSSN, cf. section 3.1), which are Semantic Pingback (cf. section 3.2) and PubSubHubbub (cf. section 3.3).

3.1 Distributed Semantic Social Network

The Distributed Semantic Social Network (DSSN) is an architecture for an open online social network which is distributed across the web. It is based on Web 2.0 and Semantic Web technologies, such as WebID for authentication, the FOAF vocabulary to describe personal profiles, Semantic Pingback (cf. section 3.2) and PubSubHubbub (cf. section 3.3) for active communication on the Semantic Web [18]. Using the DSSN architecture it is possible to communicate with `sioc:Post`s and activity feeds across server boundaries [2, 18]. This principle is used as a foundation

[2]https://www.w3.org/TR/2009/REC-xml-names-20091208/

for the proposed protocol to allow the distribution and interlinking of comments on *Web Resources*.

3.2 Semantic Pingback

Semantic Pingback[3] (cf. [17]) is an approach for bringing a social dimension to the Linked Data Web by adding semantic to the well-known Pingback mechanism (cf. [12]), a technological cornerstone of the blogosphere. By letting services notify each other about relations between RDF resources it activates simple RDF relations and makes it possible to implement communication of services and actions among edges on the Web of Data. Basically the protocol works as follows: A resource (*source*) referring to another resource (*target*) is published on the web. The Pingback *client* which is observing the *source* starts an auto discovery for the Pingback *server* announced by the *target* and sends a Pingback ping to this *server*. The ping contains the URLs of the *source* and *target* as payload. On receiving the ping, the *server* verifies it by fetching the *source* and looking for the link to the *target*. If the ping was valid, the *server* can invoke further actions, as defined by its implementation.

3.3 PubSubHubbub

PubSubHubbub is a publish subscribe protocol for the web (cf. [8]). The basic artifact is a *topic*, which can be any kind of resource on the web identified by a URL. To allow the distribution of events happening on the *topics* a *hub* is the basic dispatcher. A *publisher* of a *topic* notifies the *hub* about updates on its *topics*, while the *hub* is responsible of informing the *subscribers* by calling their specified *callback* URLs.

3.4 Linked Data

Since Linked Data is a basis of the Semantic Pingback protocol (cf. section 3.2) and the Structured Feedback protocol makes use of this paradigm, too, we provide some words on our interpretation. The four rules as formulated in [3] are: (1) Use URIs as names for things (2) Use HTTP URIs so that people can look up those names. (3) When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL) (4) Include links to other URIs[,] so that they can discover more things.

The Structured Feedback protocol relies on those rules, by assuming that all resources are identified using an HTTP URI and it is possible to query the URI of a *Web Resource*, *Comment Resource* and *Comment Container*, and RDF data is returned, either directly or by content-negotiation using HTTP `Accept`-Headers. Further links are used to connect the resources and services, and all resources are published according to these rules.

One issue we were facing with Linked Data was that nothing is mentioned about retrieving named graphs. According to the RDF Recommendation [5]: "Nevertheless, it is sometimes desirable to work with multiple RDF graphs while keeping their contents separate" and "An RDF dataset is a collection of RDF graphs. All but one of these graphs have an associated IRI or blank node. They are called named graphs, and the IRI or blank node is called the graph name. The remaining graph does not have an associated IRI, and is called the default graph of the RDF dataset." [5]. In our protocol we want to implement the *Comment Container* using named graphs to represent a feed resp. collection of *Comment Resources* related to a *Web Resource*. In order to be able to serve those sets as Linked Data, without the need of a querying service, we've decided to extend the interpretation of the Linked Data rules to support named graphs. The retrieval of statements is further discussed in the section "Browsable graphs" in [3]. Especially "describing a node" is defined as:

1. Returning all statements where the node is a subject or object; and
2. Describing all blank nodes attached to the node by one arc.

When applying those rules to an RDF dataset, we have interpreted these two rules to be executed on the default graph[4], while we have extended these two rules by the following two rules to target the other named graphs of an RDF dataset:

3. If a named graph with the same URI, as the node, as graph name is defined and accessible, returning all quad-statements of the identified named graph; and
4. All quad-statements of a named graphs with a blank node as graph name, where the blank node is contained in the set of blank nodes following the second rule, are returned.

The suggested additional rules, are interpreted in a fully backward compatible way, which means, only if a serialization format with RDF dataset support is accepted by the requesting client, the quad-statements are returned.

4. THE STRUCTURED FEEDBACK ARCHITECTURE AND PROTOCOL

The proposed approach, *Structured Feedback*, is an open protocol, distributed across multiple communicating components. It is based on several Semantic Web and DSSN protocols. Especially it is following the Linked Data paradigm (cf. section 3.4) and applies the Semantic Pingback (cf. section 3.2) and PubSubHubbub (cf. section 3.3) protocols. All components interacting through the protocol are depicted in fig. 1.

Web Resource.

Any resource published on the WWW. It can be represented in any format, but has to be identifiable and retrievable by a URL. It is encouraged to use resources in an RDF format, but baseline functionality is possible for other types as well. The *Web Resource* has to announce a *Semantic Pingback Service* and a *Comment Container* as shown in listing 1, or using the according header link relations `rel="pingback"` and `rel="alternate"`.

```
dbpedia:Leipzig
  pingback:to <http://pingback.feedback.aksw.org/> ;
  sioc:feed fb-feed:dbpedia-Leipzig .
```

Listing 1: A minimal example of the necessary triples of a *Web Resource* to announce a *Semantic Pingback Service* and a *Comment Container*

[3]http://aksw.org/Projects/SemanticPingback

[4]This interpretation should be inline with the non-normative section "Content Negotiation of RDF Datasets" of the RDF Recommendation [5]

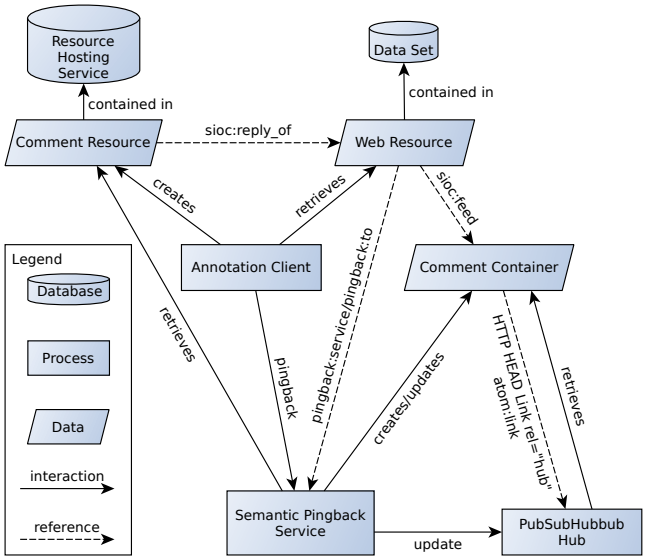

Figure 1: An architectural overview of the participating components, their interactions and references

Comment Resource.

An RDF Resource of the type `sioc:Item` representing any kind of comment or feedback related to a *Web Resource*. The *Comment Resource* is linked to the commented resource using a `sioc:reply_of` relation. It is possible to create *Comment Resources* for different kinds of feedback:

- a simple plain text contribution to a discussion using the type `sioct:Comment` as shown in listing 2,
- a structured patch as proposal to improve the quality of the *Web Resource*, e. g. by creating an instance of `cs:ChangeSet` as shown in listing 3, or
- any other resource type, which can be interpreted by the publisher of the *Web Resource*, e. g. using the RDF Review Vocabulary[5] from revyu.com.

```
fd-res:4ez-comment a sioct:Comment ;
  sioc:reply_of dbpedia:Leipzig ;
  foaf:maker <http://aksw.org/NatanaelArndt> ;
  sioc:created_at "2016-01-19T09:55:25.470Z"^^xsd:dateTime ;
  sioc:content "I'd like to ask ..." .
```

Listing 2: An example of a *Comment Resource* containing a simple text contribution (`sioc:content`) for a *Web Resource*

```
fb-res:2ug-patch a sioc:Item, cs:ChangeSet ;
  sioc:reply_of dbpedia:Leipzig ;
  cs:subjectOfChange dbpedia:Leipzig ;
  foaf:maker <http://aksw.org/NatanaelArndt> ;
  cs:creatorName "Natanael Arndt" ;
  sioc:created_at "2016-01-22T09:55:25.470Z"^^xsd:dateTime ;
  cs:createdDate "2016-01-22T09:55:25.470Z"^^xsd:dateTime ;
  cs:changeReason "Revised Triple ..." ;
  cs:addition [ a rdf:Statement ;
    rdf:subject dbpedia:Leipzig ;
    rdf:predicate ex:propA ;
    rdf:object "Hello" ;
  ] ;
  cs:removal [ a rdf:Statement ;
    rdf:subject dbpedia:Leipzig ;
    rdf:predicate ex:propA ;
    rdf:object "Hey There" ;
  ] .
```

Listing 3: An example of a *Comment Resource* containing a commit and a revision proposed as a patch for a *Web Resource*

Annotation Client.

An application providing an interface for a user to create a *Comment Resource* related to a given *Web Resource* and store it in a *Resource Hosting Service*. The *Annotation Client* can be shipped together with the *Web Resource*, e. g. as a JavaScript Application, a third party Web Service, or a stand alone client application. Depending on the implementation, the *Annotation Client* is also responsible of acting as a Pingback Client and triggering a Semantic Pingback ping to the *Semantic Pingback Service*, advertised by the commented *Web Resource*.

Semantic Pingback Service.

The *Semantic Pingback Service* implements the Semantic Pingback protocol (cf. section 3.2) and accepts ping requests. On arrival of a valid ping it is responsible for arranging an update of the *Comment Container*. This service further implements the publisher role of the PubSubHubbub protocol (cf. section 3.3). If updates are available on the container it sends an update notification to the *PubSubHubbub Hub*. The roles of the *Semantic Pingback Service* might as well be split into separate controllers responsible of receiving the Semantic Pingback ping, updating the *Comment Container* and notifying the *PubSubHubbub Hub*. This would result in a high flexibility for distribution and federation, but highly depends on the implementation and is out of scope for specification of this protocol.

Comment Container.

To be able to keep all comments on one *Web Resource* in a single container the *Comment Container* was added. It is comparable to the roll of an RSS-Feed (cf. [1]), especially to a comment feed, as it is published by some weblogs. The *Comment Container* is the item which is published to the *PubSubHubbub* hub. Users can subscribe to the container if they want to follow the comments on a *Web Resource*. Listing 4 shows an example of a *Comment Container*.

```
fb-feed:dbpedia-Leipzig a sioc:Container ;
  atom:link <http://hub.feedback.aksw.org/> ;
  sioc:container_of fb-res:4ez-comment , fb-res:2ug-patch .

fb-feed:dbpedia-Leipzig {
  fb-res:4ez-comment a sioct:Comment;
    sioc:reply_of dbpedia:Leipzig ;
    foaf:maker <http://aksw.org/NatanaelArndt> ;
    sioc:created_at
      "2016-01-19T09:55:25.470Z"^^xsd:dateTime ;
    sioc:content "I'd like to ask ..." .
```

[5] http://purl.org/stuff/rev#

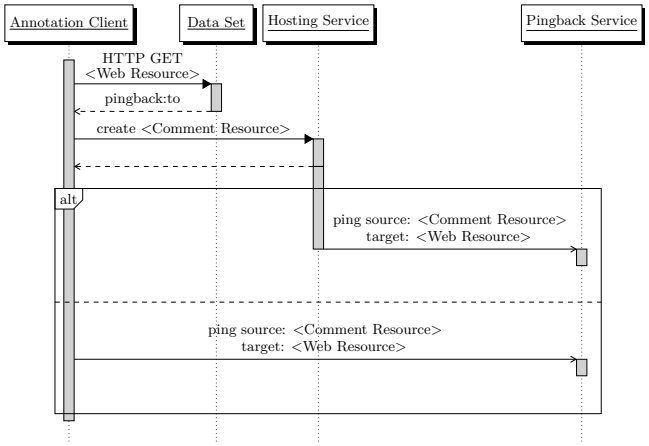

Figure 2: Sequence diagram of the initiation of the protocol and the creation of a *Comment Resource* by an *Annotation Client*

```
  fb-res:2ug-patch a sioc:Item, cs:ChangeSet ;
    sioc:reply_of dbpedia:Leipzig ;
    cs:subjectOfChange dbpedia:Leipzig ;
    foaf:maker <http://aksw.org/NatanaelArndt> ;
    cs:creatorName "Natanael Arndt" ;
    sioc:created_at
      "2016-01-22T09:55:25.470Z"^^xsd:dateTime ;
    cs:createdDate
      "2016-01-22T09:55:25.470Z"^^xsd:dateTime ;
    cs:changeReason "Revised Triple ..." ;
    cs:addition [ a rdf:Statement ;
      rdf:subject dbpedia:Leipzig ;
      rdf:predicate ex:propA ;
      rdf:object "Hello" ;
    ] ;
    cs:removal [ a rdf:Statement ;
      rdf:subject dbpedia:Leipzig ;
      rdf:predicate ex:propA ;
      rdf:object "Hey There" ;
    ] .
}
```

Listing 4: An example of a *Comment Container* containging two *Comment Resources*, which are partially mirrored in a named graph

PubSubHubbub Hub.

The *PubSubHubbub Hub* is an implementation of the hub role of the PubSubHubbub protocol (cf. section 3.3). It manages subscriptions from agents interested in notifications upon updates of the *Comment Container*.

4.1 Procedure of the Structured Feedback Protocol

A sequential overview of the protocol is given in figs. 2 and 3. The protocol is initiated by the *Annotation Client* in fig. 2, which provides a means for a user to create a comment for any *Web Resource*. The *Annotation Client* retrieves the *Web Resource* from its *Data Set* and looks for a `pingback:to`[6]

[6]It is recommended to us the simple Semantic Pingback protocol instead of the XML/RPC based protocol.

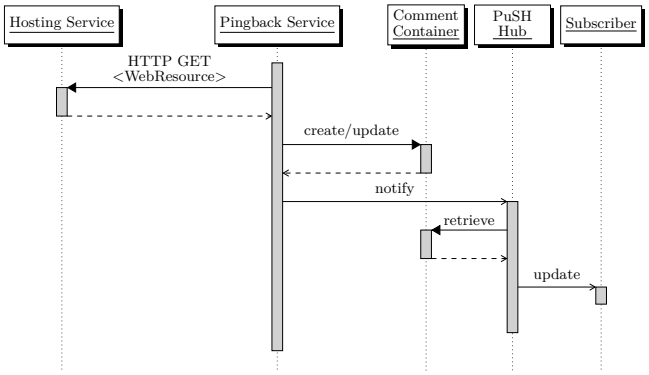

Figure 3: Sequence diagram of the federation of the *Comment Resource* by the *Pingback Service* and *PubSubHubbub (PuSH) Controller*

or `pingback:service` relation to the *Semantic Pingback Service* in charge of the requested resource. The user creates a *Comment Resource* using the *Annotation Client*, which has to contain a `sioc:reply_of` relation to the commented *Web Resource*. The *Annotation Client* then takes care of storing the *Comment Resource* in a *Resource Hosting Service* from where it is available as Linked Data. A *Resource Hosting Service* can be any SPARQL service or even a publicly available static webpage hoster, which is writable by the *Annotation Client*, e. g. through access delegation. After the *Comment Resource* was published the *Annotation Client* notifies the *Semantic Pingback Service* by sending a pingback request with the *Comment Resource* as source and the commented *Web Resource* as target. Depending on the implementation this task can also be completed by the *Hosting Service*, but to keep the selection of a *Hosting Service* as flexible as possible it is a good idea to hand over this task to the *Annotation Client*. When the pingback request was received by the *Pingback Service* it retrieves the *Comment Resource* from the *Hosting Service* for verification of the ping, as depicted in fig. 3. When the *Comment Resource* was verified by the *Semantic Pingback Service* it updates the corresponding *Comment Container* of the *Web Resource*. After the *Comment Container* was updated the *Semantic Pingback Service* (Publisher) notifies the *PubSubHubbub Hub* of the update, which in turn retrieves the updated *Comment Container* (Topic) and informs its subscribers about the update.

5. IMPLEMENTATION

We have built a reference implementation in order to validate the interoperability of the components and verify the viability of the protocol. In the following section we discuss some design decisions and problems, which occurred during the implementation.

5.1 Annotation Client

The *Annotation Client* is a piece of software that enables a user to create comments or structured patches on *Web Resources*. The actual implementation of the *Annotation Client* is a JavaScript website extension. It identifies the resource URI of the current HTML representation by looking for the relation links `foaf:primaryTopic`, `alternate` and `describedby` in the header. Then it requests the respective

RDF resource as RDF/JSON[7] (cf. [6]) formatted structured data. Additionally the URI of the *Semantic Pingback Service* is retrieved, the URL of the *Resource Hosting Service* is pre-configured in the *Annotation Client*, but should be configured by the user in the future.

The *Annotation Client* presents two dialogs (cf. section 6). At invocation it prompts a simple form for plain comments. To identify the author of the feedback, a URI to the WebID or any other personal web resource is requested. In this implementation the URI needs to be entered by the user himself using the respective input in the frontend. For future implementations the user identity could be obtained using OAuth or could be provided by the including platform, on which the extension is running. A second dialog, invocable by the user provides an editable list of all properties of the currently selected resource to the user. It provides basic property editing, deletion and insertion operations. In this dialog, the user as well has to provide an identifying URI and additionally a commit message.

The *Annotation Client* generates upon changes, deletions and insertions done by the user, changesets according to the changeset vocabulary [19], to generate a patch resource (cf. section 4). The presented patch structure in [19] is expressed using reification, where added statements are expressed as instances of `rdf:Statement` and attached to the changeset using the property `cs:addition` (deletions are handled accordingly). Additionally a reference to the commented *Web Resource* is provided using a `sioc:reply_of` relation, together with some metadata such as date and author. Due to a lag of reliable parser and serializer libraries capable of processing RDF datasets, the implementation sends string concatenated N-Quads as serialization format.

After the resource was successfully written to the *Resource Hosting Service* the *Annotation Client* directly sends the pingback request to the *Semantic Pingback Service*. This behavior is in contras to the specification in [17], where the pingback request would be sent by the *Resource Hosting Service*. We have decided to implement it in this way to keep the requirements towards a *Hosting Service* as minimal as possible and increase the selection for a service as flexible as possible, to allow a more decentralized infrastructure in the end. The source code for the *Annotation Client* is available on GitHub[8].

5.2 Resource Hosting Service

For the reference implementation of the architecture we have created a simple read/write hosting platform for RDF resources and RDF graphs in Python, inspired by the Linked Data Platform [16]. To demonstrate the simplicity we have implemented a REST interface which understands HTTP `GET`, `POST` and `PUT` requests. It is designed around our extended interpretation of the Linked Data paradigm (cf. section 3.4). To conform to the protocol, a `GET` request on a resource URI results in an RDF description of the requested resource coming from the default graph and if available the named graph with this URI.

Before a resource is sent to the *Resource Hosting Service*, the client executes a `GET` request to the services base URL to request an available URI. This request was introduced to avoid collisions of resources on the *Resource Hosting Service* as well as to have some kind of guaranty from the service, that the URI will be publicly routable on the Internet. A free resource URI is generated using a random hash and a subsequent SPARQL-ASK query to the internal store. The actual write operations are implemented as `POST` and `PUT` requests directly on the resource URI. This again makes sure that the resource will be available as Linked Data. The body of these requests has to contain the same data, which should be returned on a `GET` request to the same URI. The source code for the *Resource Hosting Service* is available on GitHub[9].

5.3 Semantic Pingback Service and Publish Subscribe System

The *Semantic Pingback Service* was implemented in PHP based on the existing generic Semantic Pingback reference implementation[10]. This implementation stores received pings in a MySQL database. The *Semantic Pingback Service* also comes with a web-form to manually send pingbacks to the service, this can help "self-publishers" of *Comment Resources* to also integrate their resources in the protocol flow. When a valid pingback was received, the controller for the management of the *Comment Container* is invoked by a callback function.

The retrieved *Comment Resource* is stored internally in a Saft[11] resp. EasyRdf[12] Triple Store with a MySQL backend. Due to the lack of widely available implementations of serializers for RDF formats with support for RDF datasets, the generation of the *Comment Container* resources and graphs was mainly done using basic string operations. To enable a flexible usage of one instance of the service for multiple data sets by providing feeds for any resource in the data set, the generation of the URI for the *Comment Containers* was implemented dynamically. Following to the host name and base path of the instance the feed URI is generated based on the according *Web Resource*. For example the feed URI for the resource `http://aksw.org/Groups/ES` is `http://feed.feedback.aksw.org/aksw.org/Groups/ES`. The PubSubHubbub publication part is done using the `php-publisher` implementation provided by the pubsubhubbub project[13].

As PubSubHubbub hub we are using Google's instance[14]. To increase the independence from central infrastructure, we also want to provide a PubSubHubbub hub implementation in the future. The source code for the *Semantic Pingback Service* is available on GitHub[15].

6. DEMONSTRATION AND EVALUATION

Based on our reference implementation of the relevant services in the protocol we have built a test setup[16]. It comprises a *Resource Hosting Service* to store *Comment*

[7]RDF/JSON was preferred over JSON-LD because it has one strait forward way of data representation, which has no need for an extra parser to work with the data.
[8]https://github.com/AKSW/AnnotationClient
[9]https://github.com/AKSW/ResourceHosting
[10]https://github.com/AKSW/SemanticPingback-Server
[11]http://safting.github.io/
[12]http://www.easyrdf.org/
[13]https://github.com/pubsubhubbub/php-publisher
[14]http://pubsubhubbub.appspot.com/
[15]https://github.com/AKSW/FeedbackServer
[16]http://feedback.aksw.org/

Resources[17] (cf. section 5.2) and a *Semantic Pingback Service*[18] (cf. section 5.3), which receives pingbacks, manages the *Comment Containers* and updates the PubSubHubbub hub. The *Annotation Client* (cf. section 5.1) is deployed on our work group homepage[19] and already increases the quality of the published data.

Using the user scripts browser extension Greasemonkey[20] for Mozilla Firefox we could also load the Annotation Client on other data sets, e. g. DBpedia[21], and include them in tests of the infrastructure as well. Figure 4 shows screenshots of the *Annotation Client* loaded on the DBpedia *Web Resource* `dbpedia:Counterparts`. Figure 4 (A) shows the dialog for entering a simple text comment to a resource (A.1), the URL entered in the top input is used to identify the author (A.2). The generated *Comment Resource* is stored as `sioct :Comment` on the *Resource Hosting Service*. In fig. 4 (B) the resource editor is displayed. It provides basic functionality to edit existing properties (B.1), to delete (B.2) and to add new properties (B.3). Similar to the simple text comment form, the user should provide a URL for identification (B.4) and additionally a commit message to describe the change (B.5).

In section 2 we have formulated some requirements which we want to evaluate now.

The first requirement was "Decentralized Storage of Comment Resources". This requirement is fulfilled firstly with the possibility of separating the *Resource Hosting Service* from the place, where the original *Web Resource* is stored. Secondly, by the openness of the protocol and the possibility to use any other *Resource Hosting Service* and manually send a pingback or use a standalone *Annotation Client* (cf. sections 5.1 and 5.2).

The second requirement was "Structured Feedback for Web Resources". This requirement is met by the usage of `cs :ChangeSet` resources (cf. [19]) to represent change proposals for resources in RDF and with the resource editor provided by the *Annotation Client*.

The third requirement was "Comment Container for Web Resource". It is fulfilled with the *Comment Containers*. A container is identifiable by a URI and can be retrieved as whole, including its items following the proposed adopted Linked Data interpretation of section 3.4, or as a single resource with references to the contained items. The *Comment Container* is referenced from the *Web Resource* with a `sioc:feed` relation.

The fourth requirement was "Active Updating of the Comment Container with new Comment Resources". It is targeted by using Semantic Pingback (cf. section 5.3). If a new *Comment Resource* is created the *Annotation Client* sends a ping with a reference to the *Comment Resource*, which then is integrated in the *Comment Container* by the *Semantic Pingback Service*.

The last formulated requirement was "Active Notification for Subscribers of Comments on Web Resources". This requirement is fulfilled by the usage of the PubSubHubbub system (cf. section 5.3). It enables interested third party agents to subscribe to the *Comment Container* of a *Web Resource*, to receive updates.

The decentralized approach of the protocol allows to distribute all of the services involved. *Resource Hosting Services* can be independently selected or deployed by a commenter, the *Pingback Service* as well as the *PubSubHubbub Hub* is selected or deployed by a resource publisher. Thus there is no single point of failure for the web wide infrastructure, only individual services can go down, as currently sometimes web sites do. Another thread to the protocol could be vandalism or spamming. This and further threads to the Semantic Web are elaborated in [20]. To protect the *Resource Hosting Services* from these threads, the maintainer of the service can install an authentication and authorization mechanism to restrict access to the service. The current implementation of our *Resource Hosting Services* already protected against adding spam messages to existing resources, be only providing a write-once mechanism. The Semantic Pingback protocol already partially contains considerations to prevent spam [17]. While still vandalism and denial of service attacks could be possible, which requires further improvements in the Semantic Pingback respective Pingback Protocols. Additionally the publisher of a *Comment Container* can implement further techniques, like spam filters before adding a *Comment Resource* to the container.

7. STATE OF THE ART AND RELATED WORK

For content providers it is often important to gather feedback from their users or to allow a discussion among the users about the presented topic. There are several possibilities for content providers to support such discussions. Generally, these possibilities resp. approaches can be categorized according to the level of distribution as follows:

1. Central to the content provider (simple commenting forms),
2. Integrated in a third party centralized online social network (Commenting Plug-Ins, Social Buttons), and
3. Web wide distributed commenting techniques.

A simple commenting form (1.) is usually already provided by webloging software or web content management systems, such as Wordpress[22], Serendipity[23], or Typo3[24]. Another option to be included into the website infrastructure of a content provider is Discourse[25]. It provides a commenting form to be integrated on a website, but unlike other third party approaches (cf. 2.) it allows website providers to decide where to host the users comments, by also providing the back-end software as Open Source. Those techniques allow users to leave a comment directly on the website of the publisher, where the comment is also stored in the domain of the content provider.

The second way of allowing user feedback on a website is the integration of an external web service or online social network (2.). This can happen by integrating social

[17] http://resource.feedback.aksw.org/
[18] http://pingback.feedback.aksw.org/ and http://feed.feedback.aksw.org/
[19] http://aksw.org/ resp. http://feedback.aksw.org/
[20] http://www.greasespot.net/
[21] http://dbpedia.org/
[22] https://codex.wordpress.org/Comments_in_WordPress
[23] http://www.s9y.org/3.html
[24] https://typo3.org/typo3-cms/key-features/complete-feature-list/
[25] http://www.discourse.org/

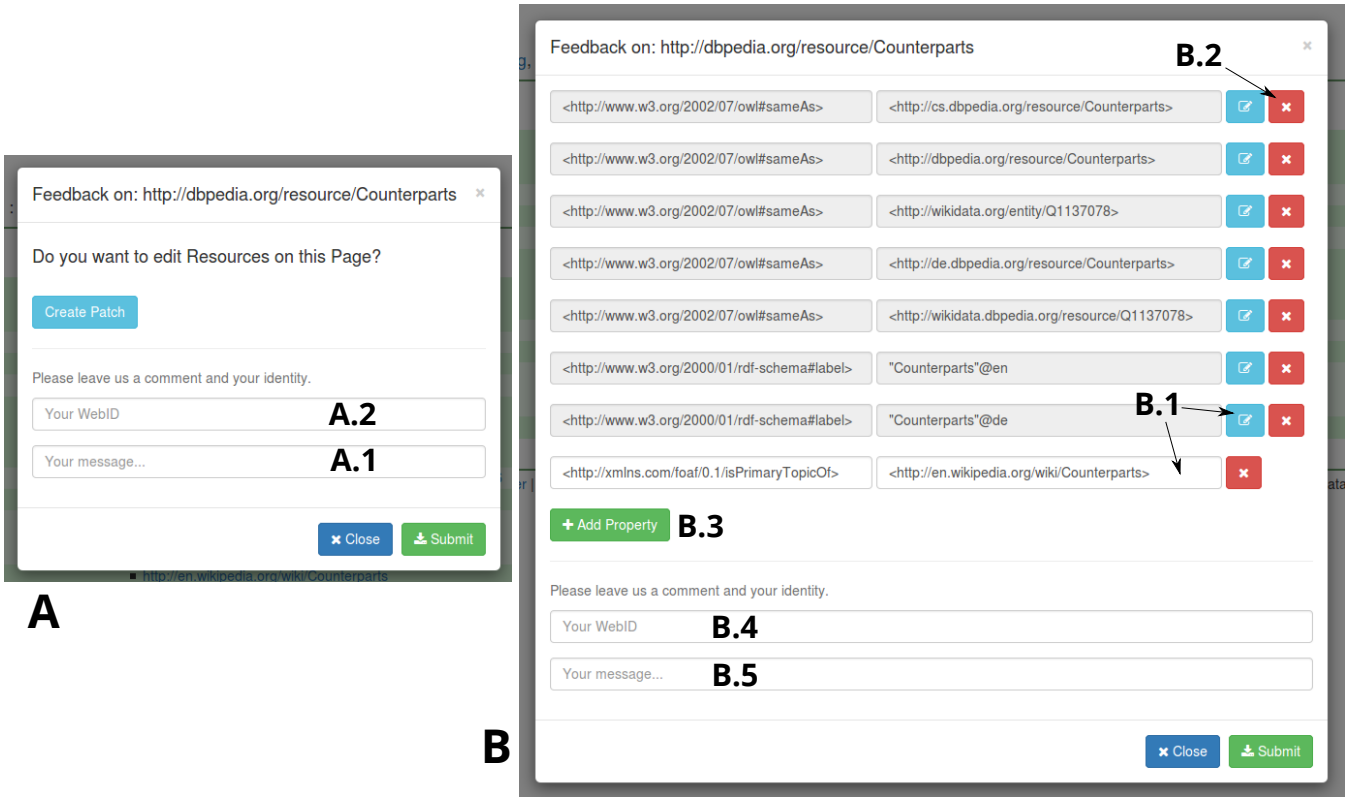

Figure 4: A: The *Annotation Client* showing the generic form for simple text comments. B: The *Annotation Client* showing the statement editor form for generating a structured commit resource to propose a patch for a *Web Resource*

buttons (Google+, Facebook like, Twitter, Reddit) to encourage users to share the link to a web resource together with a comment on an external online social network. Instead of referring the users to an external website it is also possible to directly integrate a commenting form of the external web service through a JavaScript plug-in. A very commonly used provider for this possibility is Disqus[26], besides the commenting form it provides a central server to host all the user comments and has a Facebook and Twitter integration. Facebook also provides its own Comments Plugin[27]. With this plug-in integrated into a website, users are able to leave comments on a website using their Facebook account for authentication and can share the comment on the Facebook online social network as well. All comments are stored in a central location on Facebook's servers.

Independent of the decision of the website publisher, for users it is always possible to write comments in any place on the web (3.), such as web forums, personal weblogs, review platforms, or any online social network. This results in a web wide distributed commenting system. While this is generally out of control for the content provider and it is hard to gather all feedback on a specific resource, it allows the best flexibility for the user. Techniques to target this issue are Trackback and Pingback (cf. [12] and section 3.2, implemented in Wordpress) which are used among weblogs to create back-links to other posts referring to the own post. Such protocols allow content publishers to get informed about comments which can be distributed on the web.

In respect of these three categories, the proposed Structured Feedback protocol is a crossover approach. It allows the integration of an *Annotation Client* directly into the website of the publisher to encourage users to leave comments. Also the website publisher keeps track of the comments in a *Comment Container* stored at a location of his choice. Users still have the choice whether to use the integrated client or any external commenting platform or even a standalone client. By following the proposed protocol, *Comment Resources* can be stored in any place of the web, while its still possible to inform the content publisher via Semantic Pingback. Interested third parties can subscribe to the comments on a resource using PubSubHubbub.

Sidewiki was a project introduced by Google in 2009[28], it is implemented in the Google Toolbar for Firefox and Internet Explorer, a Plug-In for Chrome or a Bookmarklet for other browsers, but is no longer available[29]. It allowed users to share annotations for web resources with other users of

[26] https://www.disqus.com/
[27] https://developers.facebook.com/docs/plugins/comments
[28] https://googleblog.blogspot.com/2009/09/help-and-learn-from-others-as-you.html
[29] https://google.com/sidewiki/, HTTP Status Code 404

the service. As it seems it relied on a central infrastructure, which was discontinued after just two years in 2011[30].

The Web Annotation Working Group at the World Wide Web Consortium[31] [13, 14] is aiming at providing a decentralized system for annotating web content. The proposed system aims at becoming comprehensive in targeting the problem of annotations on the web. Currently it provides drafts for a data model, a protocol specification and the *FindText API specification*. Further the protocol and data model also try to be flexible in the scope of supported resource types: "It will allow anyone to annotate anything anywhere, be it a web page, an ebook, a video, an image, an audio stream, or data in raw or visualized form"[32]. The protocol highly relies on the Linked Data Platform [16] for storage and retrieval of annotations (*Comment Resources*). An annotation builds a relation between a body and target, where the target is the annotated *Web Resource* and the body is the actual *Comment Resource*. The body can be any kind of resource, e.g. a text, a tag, or a movie, but also external resources, which could include change sets or patches as used in our approach.

The protocol flow of Web Annotations is analogous to our proposed protocol, while we mainly concentrate on propagating feedback in the form of patches to Linked Data Resources, the Web Annotations try to find a more general solution for arbitrary resources on the web. Similar to our approach the distribution of the protocol and the sovereignty of the resource and annotations publishers is a main requirement. "Web annotations can be linked, shared between services, tracked back to their origins, searched and discovered, and stored wherever the author wishes; the vision is for a decentralized and open annotation infrastructure"[32]. Based on the big overlap in the targeted problems, both approaches can be seen as a contribution to the same discussion about a solution for the problem from different perspectives. Where we contribute the glue part between social semantic web and distributed annotation. While both approaches can complement each other and maybe also interoperate in the in tackling this issue in the sense of a heterogeneous WWW.

Besides the possibility of leaving a simple text comment for a publisher, revyu.com (cf. [10, 11]) is a reviewing and rating web site, which allows users to create comments as machine-readable RDF metadata for the Semantic Web. The site is not only constrained to a specific set of resources, which can be reviewed, but allows reviews for any kind of resources. Using RDF as format with the RDF Review Vocabulary[5] enables a flexible expression of reviews, including ratings, useful votes and links to external data sources. Additionally, it enables versatile queries via SPARQL. [10, 11] further discusses the integration of external data sets, the usage of a semantic tagging system and the general issues of the implementation for the Semantic Web. Even though one can argue, that revyu.com is just another third-party reviewing system, it was able to improve the situation of commenting and reviewing arbitrary *Web Resources* by providing a structured user friendly and as well machine readable reviewing system.

Our approach can be seen as a further advancement of [10, 11] by combining a system of structured comments and reviews using RDF on the one hand, with the techniques of a Distributed Semantic Social Network (cf. [18], section 3.1) to target the issues of web wide distributed comments on the other hand.

8. CONCLUSION AND FUTURE WORK

In this paper, we have presented a protocol for giving feedback to resources on the Web of Data. The protocol is distributed in a way, that resource publishers and comment authors can independently decide on, where to host their infrastructure and especially resources. Using Semantic Pingback, publishers are informed about newly created comments on the web and an aggregated *Comment Container* is created as comment feed for a *Web Resource*. Further with PubSubHubbub it even enables third party agents to subscribe their services to the feed and follow newly created comments (*Comment Container*). The formulated requirements, "Decentralized Storage of Comment Resources", "Structured Feedback for Web Resources", "Comment Container for Web Resource", "Active Updating of the Comment Container with new Comment Resources" and "Active Notification for Subscribers of Comments on Web Resources" are met by the protocol and implementation (cf. section 6). We have deployed the system on our workgroup homepage[19] to demonstrate its functionality and to increase the quality of the published data.

The proposed protocol is integrated in the DSSN and a web wide infrastructure of social services. Further with the establishment of such a feedback protocol, data curation processes as well as web wide distributed co-evolution strategies for linked data sets would be promoted. As future work the *Comment Containers* should be integrate in resource curation platforms, such as OntoWiki [9] to make use of crowed based collaborative quality assessment and quality cleansing processes. This can happen by presenting the proposed changes in a curation process, where the publisher of the commented *Web Resource* can decide on applying the patch on the resource or ignore it. Another possibility would be to automatically evaluate all proposed patches and include them into a draft version of the resource, similar to the unsighted version of articles in the German Wikipedia[33]. If multiple changes target the same resource and lead to conflicts in their application, semiautomatic conflict resolution mechanism an be applied, for example as part of a co-evolution system.

The *Annotation Client* will be further develop to support the inclusion into arbitrary web pages, without causing side effects in the CSS-styling or JavaScript execution. We have already got in touch with the DBpedia maintainers to integrate our system on dbpedia.org. This would enable a wide deployment and usage of the protocol.

[30] https://en.wikipedia.org/wiki/Google_Sidewiki
[31] https://www.w3.org/annotation/
[32] from https://www.w3.org/annotation/
[33] https://de.wikipedia.org/wiki/Wikipedia:Gesichtete_Versionen (German)

9. ACKNOWLEDGEMENT

This work was partly supported by the following grants from the German Federal Ministry of Education and Research (BMBF) for the LEDS Project under grant agreement No 03WKCG11C and the European Union's Horizon 2020 research and innovation programme for the SlideWiki Project under grant agreement No 688095.

References

[1] Rss 2.0 specification. http://www.rssboard.org/rss-specification, Mar. 2009.

[2] N. Arndt and S. Tramp. Xodx: A node for the distributed semantic social network. In M. Horridge, M. Rospocher, and J. van Ossenbruggen, editors, *Proceedings of the ISWC 2014 Posters & Demonstrations Track*, volume Vol-1272 of *CEUR Workshop Proceedings*, pages 465–468, Riva del Garda, Italy, Oct. 2014.

[3] T. Berners-Lee. Linked Data. http://www.w3.org/DesignIssues/LinkedData.html, June 2009.

[4] C. Bizer, A. Jentzsch, and R. Cyganiak. State of the LOD Cloud. http://www4.wiwiss.fu-berlin.de/lodcloud/state/, March 2011.

[5] R. Cyganiak, D. Wood, and M. Lanthaler. Rdf 1.1 primer. https://www.w3.org/TR/rdf11-concepts/, Feb. 2014.

[6] I. Davis, T. Steiner, and A. J. L. Hors. Rdf 1.1 json alternate serialization (rdf/json). https://www.w3.org/TR/rdf-json/, Nov. 2013.

[7] S. Fernández. Rdfohloh, a rdf wrapper of ohloh. In *ISWC2008 workshop on Social Data on the Web (SDoW2008)*, October 2008.

[8] B. Fitzpatrick, B. Slatkin, M. Atkins, and J. Genestoux. Pubsubhubbub core 0.4 – working draft. https://pubsubhubbub.github.io/PubSubHubbub/pubsubhubbub-core-0.4.html, Feb. 2014. work in progress, expired.

[9] P. Frischmuth, M. Martin, S. Tramp, T. Riechert, and S. Auer. OntoWiki—An Authoring, Publication and Visualization Interface for the Data Web. *Semantic Web Journal*, 6(3):215–240, 2015.

[10] T. Heath and E. Motta. Revyu.com: A reviewing and rating site for the web of data. In K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux, editors, *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, page 895–902. Springer Berlin Heidelberg, 2007.

[11] T. Heath and E. Motta. Revyu: Linking reviews and ratings into the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):266–273, 2008. Semantic Web Challenge 2006/2007.

[12] S. Langridge and I. Hickson. Pingback 1.0. http://www.hixie.ch/specs/pingback/pingback, 2002.

[13] R. Sanderson. Web annotation protocol. https://www.w3.org/TR/2015/WD-annotation-protocol-20150702/, July 2015. work in progress.

[14] R. Sanderson, P. Ciccarese, and B. Young. Web annotation protocol. https://www.w3.org/TR/2015/WD-annotation-model-20151015/, Oct. 2015. work in progress.

[15] M. Schmachtenberg, C. Bizer, and H. Paulheim. Adoption of the linked data best practices in different topical domains. In P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. A. Knoblock, D. Vrandecic, P. T. Groth, N. F. Noy, K. Janowicz, and C. A. Goble, editors, *Semantic Web Conference (1)*, volume 8796 of *Lecture Notes in Computer Science*, pages 245–260. Springer, 2014.

[16] S. Speicher, J. Arwe, and A. Malhotra. Linked data platform 1.0. https://www.w3.org/TR/ldp/, Feb. 2015.

[17] S. Tramp, P. Frischmuth, T. Ermilov, and S. Auer. Weaving a Social Data Web with Semantic Pingback. In P. Cimiano and H. Pinto, editors, *Proceedings of the EKAW 2010 - Knowledge Engineering and Knowledge Management by the Masses; 11th October-15th October 2010 - Lisbon, Portugal*, volume 6317 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 135–149, Berlin / Heidelberg, October 2010. Springer.

[18] S. Tramp, P. Frischmuth, T. Ermilov, S. Shekarpour, and S. Auer. An Architecture of a Distributed Semantic Social Network. *Semantic Web Journal*, Special Issue on The Personal and Social Semantic Web, 2012.

[19] S. Tunnicliffe and I. Davis. Changeset Vocabulary. http://purl.org/vocab/changeset/schema#, May 2009.

[20] M. Vander Sande, S. Coppens, D. Van Deursen, E. Mannens, and R. Van De Walle. The terminator's origins or how the semantic web could endanger humanity. In *What will the Semantic Web look like 10 years from now? in conjunction with the ISWC2012 (SW2022)*, Nov. 2012.