Maximilian Eibl, Martin Gaedke. (Hrsg.): INFORMATIK 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 1873

A Method for Distributed and Collaborative Curation of RDF Datasets Utilizing the Quit Stack

Natanael Arndt[1] and Norman Radtke[2]

Abstract: Knowledge engineering is becoming more and more important and collaborative approaches are promising. Especially in science, collaborative knowledge engineering on research data is a key factor for success. We propose a three layered method for distributed collaboration in curation of RDF datasets with the aim to bring domain experts into the role to command the process. The method builds on the existing infrastructure and work-flows of software engineering. By adding an RDF layer on top of the Git infrastructure, the method is flexible in its adaption to various domains using domain specific editing interfaces.

Keywords: distributed collaboration; collaborative curation; quit; git; SPARQL Update; rdf dataset; domain specific; semantic web; distributed version control system; knowledge engineering

1 Introduction

Experts from various domains are working with knowledge engineering tools for organizing their domain knowledge. Collaboration, especially in science, is a key factor for successfully gaining value from the data. A need for distributed models to collaborate on common knowledge bases is recognizable in various domains. Examples are projects from the e-humanities, the *Pfarrerbuch*[3], the *Professorial Career Patterns of the Early Modern History* project[4] [Ri10], and the *Heloise – European Network on Digital Academic History*[5] [RB16]. In libraries meta-data of more and more electronic resources is gathered and shared among stakeholders. The *AMSL*[6] project is looking for collaborative curation and management of electronic resources as Linked Data [Ar14, Na14]. Also in Life Science a need for sharing data between researchers is recognizable, especially on the way to Big New Biology [TP11]. Even businesses have a need for managing data in distributed setups. In the *LUCID – Linked Value Chain Data*[7] project [Fr16] the communication of data along supply chains is subject

[1] AKSW, University of Leipzig, Augustusplatz 10, 04109 Leipzig, Germany arndt@informatik.uni-leipzig.de
[2] AKSW, University of Leipzig, Augustusplatz 10, 04109 Leipzig, Germany radtke@informatik.uni-leipzig.de
[3] `http://aksw.org/Projects/Pfarrerbuch`
[4] `http://catalogus-professorum.org/projects/pcp-on-web/`
[5] `http://heloisenetwork.eu/`
[6] `http://amsl.technology/`
[7] `http://www.lucid-project.org/`

doi:10.18420/in2017_187

of research, and in the *LEDS – Linked Enterprise Data Services*[8] project there is a need for distributed collaboration on datasets to organize business procedures.

Thessen et al. state that the three major challenges in Life Science, as they are transforming into more data-centric disciplines, are: 1) lack of comprehensive standards; 2) lack of incentives for individual scientists to share data; 3) lack of appropriate infrastructure and support [TP11]. As of our experience in e-humanities, libraries, and business the same challenges apply to these and other domains as well. The first challenge can be approached by Semantic Web and Linked Data technology [TP11], which are already adopted in various domains and appropriate domain vocabularies exist[9]. The second challenge leads to the general debate on Open Data in science which is ongoing [TP11]. The third challenge is a problem where we can and want to contribute to by providing an approach for a method to collaboratively curate datasets in a distributed setup. We are facing two problems, when multiple experts want to collaborate on creating and curating semantic data. The synchronization of the data on the technical level has to be ensured on the one hand. On the other hand, user interfaces adopted to the domain and appropriate for the regarding audience are needed. In the following we refer to the regarding audience as *domain experts*. *Domain experts* are experts in their field, which usually does not include Semantic Web. Thus in this paper we are focusing on filling the gap between the technical synchronization between the data and an interface usable by *domain experts*. We propose a three layered architecture which connects flexible RDF based user interfaces through a SPARQL interface with a Git repository and the whole Git infrastructure for distributed collaboration.

The paper is structured as follows. The state of the art is presented and discussed in Sect. 2. Our proposed setup and approach with the three layers: the domain specific layer, the store layer, and the repository layer is presented in Sect. 3. An exemplary application of the system stack is demonstrated in Sect. 4. Finally the proposed stack is discussed together with a prospect to future work in Sect. 5.

2 State of the Art

The Linked Open Data paradigm consists of four rules for making data accessible on the Web[10]. Different technologies were proposed for collaborating on Linked Data, such as *Structured Feedback* [Ar16] and the *Linked Data Platform* (LDP) respectively based on LDP the notification protocol *Linked Data Notifications* (LDN)[11] [Ca17]. The main aspect here is the resource centric collaboration approach. A Linked Data resource is under the management of a central authoritative instance, while contributors and comments can be distributed over the Internet and Web. For changing the central resource, contributors need to gain write access to it. Forking and merging resources in the Linked Data paradigm

[8] `http://www.leds-projekt.de/`
[9] Linked Open Vocabularies (LOV): `https://lov.okfn.org/`
[10] Linked Data: `http://www.w3.org/DesignIssues/LinkedData.html`
[11] `https://www.w3.org/TR/ldn/`

would include moving data to a different namespace, which would involve breaking the context of the data, especially incoming links.

Curation systems, user interfaces and editors can give different kinds of contributors the possibility to editing RDF data. Semantic Data Wiki systems, such as OntoWiki[12] [Fr15, FAM16, KVV06] and RDF editors such as WebProtégé [Tu13] allow mainly people familiar with the RDF data model to collaborate. Such Wiki systems provide a platform to manage and edit datasets in a centralized place. Customizable form generation tools, for example RDForms[13], can be used to adapt the user interface directly to the needs of a domain expert, while it directly handles RDF data. With the current W3C Candidate Recommendation *Shapes Constraint Language*[14] (SHACL) a standard is developed, which can be used to create very customized user interfaces for viewing, editing and validating RDF data.

Customized RDF editing tools using RDForms and SHACL or editors like Protégé allow ontology and domain experts to locally curate data in a common RDF data model. Once a common RDF data model is established, still the RDF graphs have to be synchronized between parties to enable a collaboration process. In the field of software engineering source code editors are used for producing the individual artifacts of a program, while distributed source code management systems (DSCM) are used to synchronize the source code development process between participants. Widely used DSCMs are Git[15] and Mercurial[16]. For RDF data multiple approaches exist for allowing versioning and synchronization of datasets. The TailR [MKS15], R43ples [GHU16] and R&Wbase [Sa13] approaches provide versioning systems for tracking and exchanging the data's history, while they only provide limited or no support for branching and merging [AM17].

3 Our Setup and Approach

We have identified three layers of abstraction which can be used to create a flexible collaboration setup. The layers, as depicted in Fig. 1, are: (1) Domain Specific Layer, (2) Store Layer, and (3) Repository Layer. The top layer (1) represents the interface to domain experts with domain specific tools, which are already able to produce RDF data. This layer especially allows the system to be adapted to any domain and could also allow the collaboration between heterogeneous editors. On the lowest layer (3) the technical infrastructure for synchronizing and transporting the data on the network of participants is organized. At this level the system relies on already established and successfully used technology from the software engineering domain. The store layer (2) between the domain specific UI layer and the technical infrastructure layer provides the glue. It transforms change

[12] http://ontowiki.net/
[13] http://rdforms.org/
[14] https://www.w3.org/TR/shacl/
[15] http://git-scm.com/
[16] http://mercurial-scm.org/

operations on the RDF data to the corresponding synchronization operations on the DSCM system.

The individual layers of our setup provide interfaces to the outside and are connected with each other using respective interfaces and protocols for achieving the necessary levels of abstraction. The domain expert level provides *user interfaces* allowing to adjust the software to the respective audience. The store layer provides a *standard SPARQL Query and Update interface* allowing software, which is following this standard, to read and write the stored RDF graphs. The store layer in turn writes the RDF graphs to the Git repository using the *Git API*. The network layer is then able to synchronize the stored repository using the *Git transfer protocol*[17].

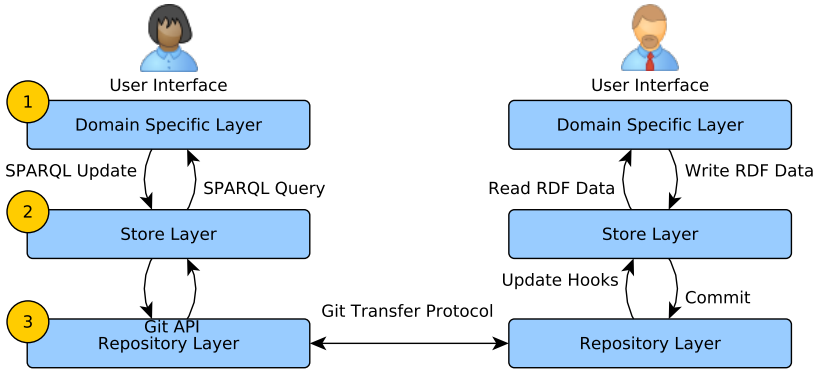

Fig. 1: The functions and flow of information

Domain Specific Layer Data models in RDF can be very complex for expressing the relations of a domain, also domain experts want to express their knowledge in complete and very complex models. But a domain expert might not be familiar with the technical structure and vocabulary of RDF. The topmost layer is the layer of domain specific user interfaces. This layer is used by domain experts providing an interface adapted to the needs of the domain and expressed in the language of the domain. Components in this layer are connected to the store layer, from where it can read and to where it can write RDF graph data. The relevant interface to be addressed by these components is SPARQL. Providing a SPARQL interface to this layer allows that already existing components, which are capable of communicating with a SPARQL endpoint, can be integrated into the system. This enables reuse of non-collaborative software in a distributed collaboration setup. Furthermore the process of selecting a proper user interface for an editor is independent from the underlaying distributed communication stack.

[17] `https://git-scm.com/book/en/v2/Git-Internals-Transfer-Protocols`

The Store Layer This layer connects to the domain specific layer on the top and to the repository layer on the bottom. This intermediary role is implemented by providing a SPARQL endpoint and the usage of the Git API. SPARQL Update operations are transformed to filesystem operations and the according Git operations. On changes in the underlaying repository layer, the RDF graphs in the store are updated accordingly. Concurrent edit operations on a dataset, which are detected on the repository layer (merge conflicts) are resolved by according merge strategies for RDF datasets on the store layer (cf. [ARM16, AM17]). Translating SPARQL operations to Git operations allows to make the Git API and transfer protocol transparent for systems connecting to the SPARQL interface.

Repository Layer Distributed collaborative curation of RDF graphs supposes that data must be available from and at different locations on the Web. Git provides the possibility to store repositories in different places on the Web, which can individually and independently evolve and be synchronized at any time using the Git transfer protocol. In each repository again the Git system allows to store multiple versioning-branches in parallel, which allows to postpone the consolidation and conflict resolution. The branching system of Git also allows operations like *Fork*[18] and *Pull Request*[19]. To organize the workflow of branching and merging for the development of an RDF vocabulary, Halilay et al. have developed the Git4Voc method [Ha16].

4 Application

For showcasing the presented method and the interplay between the individual layers, we have developed a prototype. The prototype provides an editing interface to create RDF resources, which are submitted to the SPARQL endpoint of a Quit Store [ARM16]. The Quit Store commits the changed triples to a Git repository and triggers a synchronization with a Git push operation. As depicted in Fig. 2, this stack can be setup on multiple clients simultaneously. In addition to the collaboration setup a server is subscribed for changes (web hook) on an on-line repository. As soon as new commits are submitted to the repository, a webpage presenting the content of the collaboratively created RDF graph is rendered and updated using Jekyll[20] and Jekyll-RDF[21].

The aim of the domain specific layer is to address domain experts which might not be familiar with RDF. Thus we abstract the underlaying data model by employing the form rendering library RDForms[22]. The forms are generated based on templates, which also ensure the adherence of the generated data to the applications data model. Furthermore the template

[18] Copy an entire repository into an own namespace for an independent development but keep a reference to the origin

[19] A request to merge a given branch into another branch, this can even happen across repositories

[20] Jekyll static website generation system: `https://jekyllrb.com/`

[21] RDF plugin for Jekyll: `https://rubygems.org/gems/jekyll-rdf`

[22] `http://rdforms.com/`

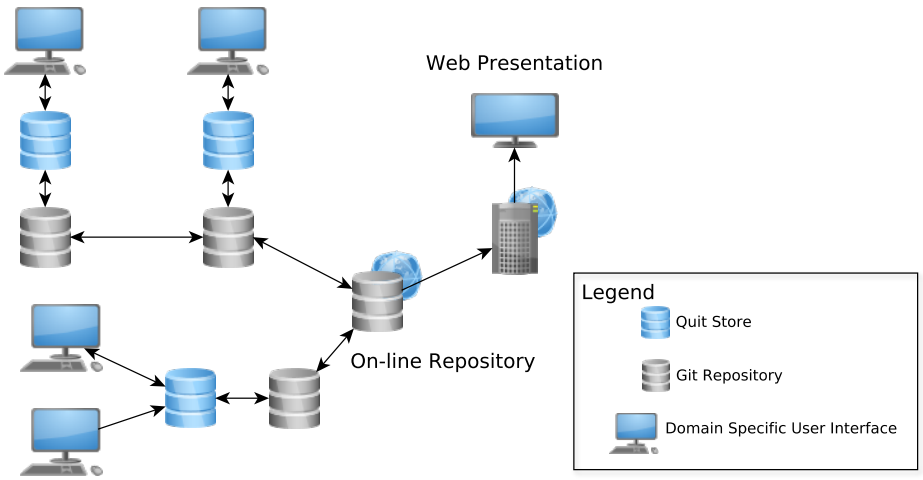

Fig. 2: A collaboration setup using the Quit Stack

based static website generation process with Jekyll-RDF creates a browsing interface on the data taking out the complexity of the RDF data model for visitors. The store layer is implemented using the Quit Store as a file based in-memory-quad-store. The Quit Store provides a SPARQL Update Endpoint, which receives the changes made using the generated forms and performs the respective transactions on the Git repository. This demonstration setup is similar to the principle of GitHub pages[23], but adapted to the collaboration and publication of RDF data. Multiple commiters can contribute to a commonly synchronized Git repository, while the consensus expressed in a specific branch is rendered to a public webpage. We have published the source code of the editing interface[24], the store layer implementation of the Quit Store[25] and the Jekyll-RDF plugin[26] as Open Source.

5 Discussion and Conclusion

With the presented three layer architecture and the associated method of distributed collaborative data creation and curation we are able to make the complexity of an RDF data model and the necessary SPARQL and Git interfaces transparent to domain experts. Also using the underlaying Git repository system it is possible to formulate feedback to the RDF datasets as pull-requests. In turn this allows multi-staged review and curation process as

[23] https://pages.github.com/
[24] https://github.com/white-gecko/vgaf
[25] https://github.com/AKSW/QuitStore
[26] https://github.com/white-gecko/jekyll-rdf

known from the *successful Git branching model* (aka. gitflow)[27] and as adapted to RDF vocabularies by Halilay et al. [Ha16]. Thinking further this as well enables workflows known from *Content Management Systems* (CMS) and would allow data based CMS workflows in Data Management System (*DMS*). Tracking all changes on the data in a non linear system as Git, in the end also tracks the provenance of the data throughout the collaborative curation process [ANM17].

With the abstraction towards the domain specific layer it is possible to use multiple editors which might provide varying advantages for the author of the data, while the editor does not need any synchronization and collaboration support. This setup can also be understood as a foundation for an Integrated Development Environment (IDE) for data. Due to the distributed character *Continuous Integration* processes can be attached to the network, as know from software engineering, for testing and verifying the data against SHACL shapes or RDFUnit test cases [MJ16]. In the future we also want to investigate the interplay between the validation and consistency check step based on SHACL shapes and the generation of edit interfaces, which can as well be based on SHACL shapes[28].

With this method we are providing the foundation to build further data management systems around a Git infrastructure for RDF datasets, as it already exists for software engineering. This allows us as computer scientists to concentrate on the architecture of the software, since the data curation process should be pursued by data scientists and domain experts rather then computer scientists and semantic web experts.

Acknowledgement

This work was partly supported by a grant from the German Federal Ministry of Education and Research (BMBF) for the LEDS Project under grant agreement No 03WKCG11C.

References

[AM17] Arndt, Natanael; Martin, Michael: Decentralized Evolution and Consolidation of RDF Graphs. In: 17th International Conference on Web Engineering (ICWE 2017). ICWE 2017, Rome, Italy, June 2017.

[ANM17] Arndt, Natanael; Naumann, Patrick; Marx, Edgard: Exploring the Evolution and Provenance of Git Versioned RDF Data. In (Fernández, Javier D.; Debattista, Jeremy; Umbrich, Jürgen, eds): 3rd Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW) co-located with 14th European Semantic Web Conference (ESWC 2017). Portoroz, Slovenia, May 2017.

[Ar14] Arndt, Natanael; Nuck, Sebastian; Nareike, Andreas; Radtke, Norman; Seige, Leander; Riechert, Thomas: AMSL: Creating a Linked Data Infrastructure for Managing Electronic

[27] http://nvie.com/posts/a-successful-git-branching-model/

[28] https://www.w3.org/TR/shacl-ucr/

Resources in Libraries. In (Horridge, Matthew; Rospocher, Marco; van Ossenbruggen, Jacco, eds): Proceedings of the ISWC 2014 Posters & Demonstrations Track. volume Vol-1272 of CEUR Workshop Proceedings, Riva del Garda, Italy, pp. 309–312, October 2014.

[Ar16] Arndt, Natanael; Junghanns, Kurt; Meissner, Roy; Frischmuth, Philipp; Radtke, Norman; Frommhold, Marvin; Martin, Michael: Structured Feedback: A Distributed Protocol for Feedback and Patches on the Web of Data. In: Proceedings of the Workshop on Linked Data on the Web co-located with the 25th International World Wide Web Conference (WWW 2016). volume 1593 of CEUR Workshop Proceedings, Montreal, Canada, April 2016.

[ARM16] Arndt, Natanael; Radtke, Norman; Martin, Michael: Distributed Collaboration on RDF Datasets Using Git: Towards the Quit Store. In: 12th International Conference on Semantic Systems Proceedings (SEMANTiCS 2016). SEMANTiCS '16, Leipzig, Germany, September 2016.

[Ca17] Capadisli, Sarven; Guy, Amy; Lange, Christoph; Auer, Sören; Sambra, Andrei; Berners-Lee, Tim: Linked Data Notifications: a resource-centric communication protocol. In: 14th European Semantic Web Conference (ESWC 2017). Portoroz, Slovenia, 2017.

[FAM16] Frischmuth, Philipp; Arndt, Natanael; Martin, Michael: OntoWiki 1.0: 10 Years of Development - What's New in OntoWiki. In: SEMANTiCS2016 Poster and Demo Track. CEUR Workshop Proceedings, Leipzig, Germany, September 2016.

[Fr15] Frischmuth, Philipp; Martin, Michael; Tramp, Sebastian; Riechert, Thomas; Auer, Sören: OntoWiki—An Authoring, Publication and Visualization Interface for the Data Web. Semantic Web Journal, 6(3):215–240, 2015.

[Fr16] Frommhold, Marvin; Arndt, Natanael; Tramp, Sebastian; Petersen, Niklas: Publish and Subscribe for RDF in Enterprise Value Networks. In: Proceedings of the Workshop on Linked Data on the Web co-located with the 25th International World Wide Web Conference (WWW 2016). 2016.

[GHU16] Graube, Markus; Hensel, Stephan; Urbas, Leon: Open Semantic Revision Control with R43Ples: Extending SPARQL to Access Revisions of Named Graphs. SEMANTiCS 2016, ACM, New York, NY, USA, pp. 49–56, 2016.

[Ha16] Halilaj, Lavdim; Grangel-González, Irlán; Coskun, Gökhan; Auer, Sören: Git4Voc: Git-based Versioning for Collaborative Vocabulary Development. In: 10th International Conference on Semantic Computing. Laguna Hills, California, February 2016.

[KVV06] Krötzsch, Markus; Vrandečić, Denny; Völkel, Max: Semantic MediaWiki. In: 5th International Semantic Web Conference, ISWC 2006. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 935–942, November 2006.

[MJ16] Meissner, Roy; Junghanns, Kurt: Using DevOps Principles to Continuously Monitor RDF Data Quality. In: 12th International Conference on Semantic Systems Proceedings (SEMANTiCS 2016). CEUR Workshop Proceedings, Leipzig, Germany, September 2016.

[MKS15] Meinhardt, Paul; Knuth, Magnus; Sack, Harald: TailR: A Platform for Preserving History on the Web of Data. In: Proceedings of the 11th International Conference on Semantic Systems. SEMANTICS '15, ACM, New York, NY, USA, pp. 57–64, 2015.

[Na14] Nareike, Andreas; Arndt, Natanael; Radtke, Norman; Nuck, Sebastian; Seige, Leander; Riechert, Thomas: AMSL: Managing Electronic Resources for Libraries Based on Semantic Web. In: Proceedings of the INFORMATIK 2014. volume P-232, pp. 1017–1026, September 2014.

[RB16] Riechert, Thomas; Beretta, Francesco: Collaborative Research on Academic History using Linked Open Data: A Proposal for the Heloise Common Research Model. CIAN-Revista de Historia de las Universidades, 19(0), 2016.

[Ri10] Riechert, Thomas; Morgenstern, Ulf; Auer, Sören; Tramp, Sebastian; Martin, Michael: Knowledge Engineering for Historians on the Example of the Catalogus Professorum Lipsiensis. In: Proceedings of the 9th International Semantic Web Conference. volume 6497 of Lecture Notes in Computer Science, Springer, Shanghai, China, pp. 225–240, 2010.

[Sa13] Sande, Miel Vander; Colpaert, Pieter; Verborgh, Ruben; Coppens, Sam; Mannens, Erik; de Walle, Rik Van: R&Wbase: git for triples. In (Bizer, Christian; Heath, Tom; Berners-Lee, Tim; Hausenblas, Michael; Auer, Sören, eds): LDOW. volume 996 of CEUR Workshop Proceedings. CEUR-WS.org, 2013.

[TP11] Thessen, Anne E.; Patterson, David J.: Data issues in the life sciences. ZooKeys, 150(150):15–51, 2011.

[Tu13] Tudorache, Tania; Nyulas, Csongor; Noy, Natalya F; Musen, Mark A: WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web. Semantic web, 4(1):89–99, 2013.