Decentralized Collaborative Knowledge Management using Git

Extended Abstract

Natanael Arndt
Agile Knowledge Engineering and Semantic Web (AKSW)
Institut fÃijr Informatik, Leipzig University
Institut fÃijr Angewandte Informatik (InfAI) e.V.
Leipzig, Germany
arndt@informatik.uni-leipzig.de

Michael Martin
Agile Knowledge Engineering and Semantic Web (AKSW)
Institut fÃijr Angewandte Informatik (InfAI) e.V.
Leipzig, Germany
martin@informatik.uni-leipzig.de

ABSTRACT

Apart from documents, datasets are gaining more attention on the World Wide Web. An increasing number of the datasets on the Web are available as Linked Data, also called the *Linked Open Data Cloud*[1] or *Giant Global Graph*[2]. Collaboration of people and machines is a major aspect of the World Wide Web and as well of the Semantic Web. Currently, the access to RDF data on the Semantic Web is possible by applying the Linked Data principles[3], and the SPARQL specification[4], which enables clients to access and retrieve data stored and published via SPARQL endpoints. RDF resources in the Semantic Web are interconnected and often correspond to previously created vocabularies and patterns. This way of reusing existing knowledge facilitates the modeling and representation of information and may optimally reduce the development costs of a knowledge base. As a result of the collaborative reuse process, structural and content interferences as well as varying models and contradictory statements are inevitable.

CCS CONCEPTS

• **Information systems → Query languages**; **Version management**; **Resource Description Framework (RDF)**; *Data management systems*; *Middleware for databases*; RESTful web services.

KEYWORDS

Versioning, RDF, Semantic Web, Git, Distributed Version Control System, Distributed Collaboration, Knowledge Engineering, Quit Store

ACM Reference Format:
Natanael Arndt and Michael Martin. 2019. Decentralized Collaborative Knowledge Management using Git: Extended Abstract. In *Companion Proceedings of the 2019 World Wide Web Conference (WWW '19 Companion), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3308560.3316523

[1] http://lod-cloud.net/
[2] http://dig.csail.mit.edu/breadcrumbs/node/215
[3] http://www.w3.org/DesignIssues/LinkedData.html
[4] https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/

WWW '19 Companion, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6675-5/19/05.
https://doi.org/10.1145/3308560.3316523

Projects from a number of domains are striving for distributed models to collaborate on common knowledge bases. In the domain of e-humanities projects often come with a need to explore and track provenance and the evolution of the domain data [6, 7]. In the context of managing historical prosopographical data, the source of the statements is relevant to evaluate their credibility and to consider the influence of their environment. In libraries, metadata of electronic library resources are gathered and shared among stakeholders to collaboratively curate and manage the resources as Linked Data [2, 5]. In a collaborative data curation setup the origin of any statement needs to be identified in order to be able to track back the conclusion of license contracts and identify sources of defective metadata. But even enterprises have a need to manage data in distributed setups to organize the communication of data along supply chains or business processes [4].

Distributed systems such as the *Solid*[5] platform as an advancement of the architecture of a distributed semantic social network provide possibilities to collaborate in a distributed network. Nevertheless, the subject of collaboration is currently kept in a central place where all contributions are incorporated; the organization of a fully decentralized collaboration process is still subject to future work. In general, currently the collaboration on Linked Data Sets is mainly done by keeping a central version of a dataset. The systems available to collaborate on Linked Data are central SPARQL endpoints and Wiki systems where collaboration happens on a single, shared instance. This central approach for a synchronized state has drawbacks in scenarios in which the existence of different versions of the dataset is preferable. Furthermore, the evolution of a dataset in a distributed setup is not necessarily happening in a linear manner. Multiple versions of a dataset occur if the participants do not all have simultaneous access to the central dataset. If a consensus on the statements in a dataset is not yet reached, multiple viewpoints need to be expressed as different versions of the dataset. Hence, a system that fosters the evolution of a dataset in a distributed collaboration setup needs to **support divergence** of datasets as asynchrony and dissent; **reconcile diverged states** of datasets; and **synchronize** different distributed derivatives of the dataset. As a consequence of the reconciliation we also needs to identify possible occurring conflicts and contradictions, and offer workflows to resolve identified conflicts and contradictions. The dimensions of *consensus* vs. *dissent* and *synchronicity* vs. *asynchrony* are depicted in fig. 1. While the *dissent*-dimension comes with the

[5] https://solid.mit.edu/

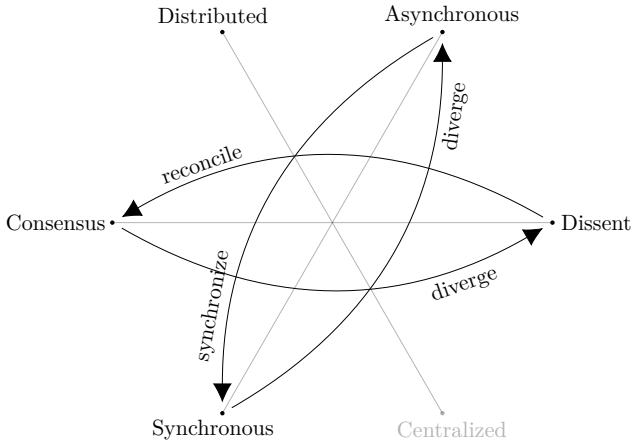

Figure 1: Collaboration can be organized *centralized* or *distributed*. When dealing with distributed collaboration, workspaces can *diverge* and the aspects *dissent* and *asynchrony* have to be considered. Even so supporting distribution, dissent, and asynchrony increases the flexibility of a collaboration process, collaboration aims for *consensus* which requires all participants to have access to a common shared knowledge, expressed in *synchronous* workspaces. Thus processes to *synchronize* and *reconcile* are needed.

collaborative character, *asynchrony* is introduced due to the *distributed* conception of our setup. Both of the dimensions can lead to a diverged state of a dataset in a collaborative curation scenario.

In the early days of computers, the term *software crisis* was coined to describe the immaturity of the software engineering process and software engineering domain. The process of creating software could be made more reliable and controllable by introducing software engineering methods. Version control is an important aspect to organize the collaborative evolution of software. Early *version control systems* (VCS), such as *CVS* and *Subversion*, allowed central repositories with a linear version history to be created. Distributed VCS (DVCS), such as *Darcs*, *Mercurial*, and *Git*, were developed to allow every member of a distributed team to fork the current state of the programs source code and individually contribute new features or bug-fixes as pull-requests. Learning from software engineering history where DVCS have helped to overcome the software crisis, we claim that adapting DVCS to Linked Data is a means to support decentralized and distributed collaboration processes in knowledge management. The subject of collaboration in the context of Linked Data are datasets instead of source code files. Similar to source code development with DVCS, individual local versions of a dataset are curated by data scientists and domain experts.

In our previously published paper [1] we present *Quit Store*, it was inspired by and it builds upon the successful Git system. The approach is based on a formal expression of evolution and reconciliation of distributed datasets. It provides support to branch, merge, and synchronize distributed RDF datasets. During the collaborative curation process, the system automatically versions the RDF dataset and tracks provenance information. The provenance information is expressed in RDF using PROV-O and can be accessed through a dedicated SPARQL 1.1 endpoint. To version the data, the system relies on the pure RDF data model and not on support for additional semantics such as OWL or SKOS. To support distributed collaboration we propose a methodology of using a Git repository to store the data in combination with a SPARQL 1.1 interface to access it. The SPARQL 1.1 interface provides an integration layer to make the collaboration features of Quit accessible to applications operating on RDF datasets. Most recently we have extended the Quit system with the Quit Editor Interface Concurrency Control [3] to support editors in managing overlapping operations. To reconcile diverged datasets a merge process is provided. The merge process is guarded by the specific merge strategies for RDF data: *Union Merge*, *All Ours/All Theirs*, *Three-Way-Merge*, and *Context Merge*. This setup can enable complex distributed collaboration strategies. As there is a big ecosystem of methodologies and tools around Git to support the software development process, the Quit Store can support the creation of such an ecosystem for RDF dataset management.

ACKNOWLEDGMENTS

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 the Federal Ministry for Economic Affairs and Energy (BMWi) for the Platona-M project under the grant number 01MT19005A, and the DFG project *Professorial Career Patterns of the Early Modern History: Development of a scientific method for research on online available and distributed research databases of academic history* under the grant agreement No 317044652.

REFERENCES

[1] Natanael Arndt, Patrick Naumann, Norman Radtke, Michael Martin, and Edgard Marx. 2018. Decentralized Collaborative Knowledge Management using Git. *Journal of Web Semantics* (2018). https://doi.org/10.1016/j.websem.2018.08.002

[2] Natanael Arndt, Sebastian Nuck, Andreas Nareike, Norman Radtke, Leander Seige, and Thomas Riechert. 2014. AMSL: Creating a Linked Data Infrastructure for Managing Electronic Resources in Libraries. In *Proceedings of the ISWC 2014 Posters & Demonstrations Track*. Riva del Garda, Italy. http://ceur-ws.org/Vol-1272/paper_66.pdf

[3] Natanael Arndt and Norman Radtke. 2019. Conflict Detection, Avoidance, and Resolution in a Non-Linear RDF Version Control System: The Quit Editor Interface Concurrency Control. In *Companion Proceedings of the 2019 World Wide Web Conference (WWW '19 Companion)*. San Francisco, CA, USA. https://doi.org/10.1145/3308560.3316519

[4] Marvin Frommhold, Natanael Arndt, Sebastian Tramp, and Niklas Petersen. 2016. 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)*. http://ceur-ws.org/Vol-1593/article-09.pdf

[5] Andreas Nareike, Natanael Arndt, Norman Radtke, Sebastian Nuck, Leander Seige, and Thomas Riechert. 2014. AMSL: Managing Electronic Resources for Libraries Based on Semantic Web. In *Proceedings of the INFORMATIK 2014: Big Data – Komplexität meistern*. Gesellschaft fÃijr Informatik e.V., Stuttgart, Germany. https://dl.gi.de/bitstream/handle/20.500.12116/2713/1017.pdf

[6] Thomas Riechert and Francesco Beretta. 2016. 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). http://e-revistas.uc3m.es/index.php/CIAN/article/view/3147

[7] Thomas Riechert, Ulf Morgenstern, Sören Auer, Sebastian Tramp, and Michael Martin. 2010. Knowledge Engineering for Historians on the Example of the Catalogus Professorum Lipsiensis. In *Proceedings of the 9th International Semantic Web Conference (ISWC2010)*. Springer, Shanghai, China. https://doi.org/10.1007/978-3-642-17749-1_15