

Schriften des Instituts für Dokumentologie und Editorik — Band 13

Versioning Cultural Objects Digital Approaches

edited by

Roman Bleier, Sean M. Winslow

2019

BoD, Norderstedt

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de/> abrufbar.

Digitale Parallelfassung der gedruckten Publikation zur Archivierung im Kölner Universitäts-Publikations-Server (KUPS). Stand 18. Dezember 2019.

2019

Herstellung und Verlag: Books on Demand GmbH, Norderstedt

ISBN: 978-3-7504-2702-0

Einbandgestaltung: Julia Sorouri and Stefan Dumont; Coverbild gestaltet von Vinayak Das Gupta.

Satz: Roman Bleier, Sean M. Winslow und Lua \TeX

Extending Versioning in Collaborative Research

Martina Bürgermeister

Abstract

In the digital world, software development is the domain in which the management of versions, also called versioning, was first introduced and became common praxis. Versioning mechanisms usually run in the backend of web environments for data security reasons. However, what is equally important is the use of versioning as a mechanism for evidence and reference for the users. When users collaboratively create research data on web platforms, a *version history* enables others to verify and reanalyse the data. Moreover, versioning mechanisms could support scholars to make research more transparent and discursive. Following this reasoning, this paper discusses versioning as performed by version control systems and investigates how this kind of versioning can be extended to serve scholarly practice.

1 Introduction

Web-based collaborative research environments in which users create, revise, and modify contributions together, make the ephemeral status of digital resources especially evident. To counteract this shortcoming, scholars need a systematic method of organizing their resources which tracks both origin and changes. One way to do this is to automatically save all content for each change as a version with a timestamp that can be made available on request. Equally important in the research process is the management of these versions, including mechanisms to persistently refer to and access individual versions. Version management enables others to verify, reanalyse, and reference the data generated with each edit, increasing the transparency of research processes.

In the digital world, software development was the domain where the management of versions, or versioning, was first introduced. Designed as a support tool during the software development process, in a versioning system all changes are logged and assigned a unique number or name in relation to the current developmental state, so that they are always ready to be recovered at a later moment. Similar versioning mechanisms run in the backend of web environments for data security reasons. Versioning management provided by Version Control Systems (VCSs) were first developed for the software industry to make production more efficient, but they have a benefit that clearly goes beyond their original use:

[...] not only does using a VCS solve many common problems when writing code, it can also improve the scientific process. By tracking your code development with a VCS and hosting it online, you are performing science that is more transparent, reproducible, and open to collaboration (Blischak et al.).

How can this benefit of VCSs be applied to research environments in the humanities? How can these systems support collaborative research? And are they good enough? Firstly, this paper will define collaborative environments in research and highlight some examples from the field of Digital Humanities (DH). Secondly, the paper will take a closer look at VCSs in general and at *Wikipedia's* adoption of a software development approach to versioning for the purposes of collaborative editing. Starting from this example it will be shown that versioning systems can be extended for research purposes and how they could be implemented. Finally, the paper will juxtapose three ontologies which are frequently used in DH and in the cultural heritage sector (FRBRoo, OAI-ORE and PROV-O), and how these ontologies approach versioning-like concepts. However, a finished implementation will not be provided in this paper.

2 Web-based collaborative research in DH

The importance of dealing with changes in digitally published web content becomes apparent especially within wiki-like research environments. For the purposes of this paper, Carusi's and Reimer's definition for such environments will be used:

To summarise, we found that the term used was not important, though the understandings associated with the terms *VRE*, *Collaboratory* and *Gateway* are converging on a set of characteristic features: an electronic web-based environment for a) access to data, tools, resources; b) co-operation or collaboration with other researchers at the same or different institutions; c) cooperation at the intra- and inter-institutional levels; or d) preserving or taking care of data and other outputs. Not all of these environments serve all of these functions, but they generally serve two or more (15).

The focus of this paper is on web-based environments where content is collaboratively generated and should be preserved and be openly available. Henceforth the term *collaboratory* will be used for such environments. Virtually anyone could be a potential collaboratory contributor: researchers, students, or interested laypersons.

The following examples of collaboratories are taken from the DH field and include, to a certain extent, a versioning strategy.

One of the earliest collaboratories in this context was *Suda On Line*,¹ an attempt to collaboratively translate the *Suda*, a Byzantine Greek historical encyclopaedia. *Suda On Line* started at the end of the 1990s and the last translation was published in 2014. At that time, the platform had about 200 contributors from more than 20 countries (Suda On Line). Registered users could edit and translate passages of the *Suda*. Administrators revised the translation or any changes made and published it on the platform. Despite not being able to retrieve older versions, a basic type of version history is presented: it is possible to know who made the entry, who vetted the transcription, and when it was written.

Another example is *Papyri.info*,² which was launched in 2006. *Papyri.info* aggregates papyrological material from different databases and makes them available and describable. Users of *Papyri.info* can browse through the distributed resources and, when registered, add new descriptions or change existing ones. There is also a peer-review process for the edits. The whole project is managed using *Git*:³ all edits are thus recorded, versioned, and recoverable. *Papyri.info* provides a full editorial history by linking to the software development platform project site.

*Annotated Books Online (ABO)*⁴ is a virtual research environment for scholars and students. It was launched in 2013 and is part of the research project *A Collaboratory for the Study of Reading and the Circulation of Ideas in Early Modern Europe*. *ABO* contributes to the study of marginalia and enables the tracing of reading practices and the use of books (Visser 67–69). Registered users of *ABO* can also transcribe and translate the marginalia. All transcriptions and translations are supervised by the project administrators, who guarantee the quality of the contributions. Each contributor can re-edit an already edited area. There is a basic edit history, which displays who created the entry and when the record was last modified.

The last collaboratory⁵ considered here is *Monasterium.net*.⁶ It was founded in 2001 and it is Europe's largest collaborative archive for charters from the middle ages and the early modern period. *Monasterium.net* allows diplomatists, archivists, and even interested laypersons to share their research with the general public. *Monasterium.net* does not provide any version history and former versions can no longer be displayed, but it does enable different versions to coexist as interpretations of the archival

¹ www.stoa.org/sol.

² www.papyri.info.

³ *Git* is one of the most used version control systems at the moment. The repository of the *papyri.info* editing framework was initiated in 2008 under: github.com/papyri/sosol/graphs/contributors.

⁴ www.annotatedbooksonline.com

⁵ For more on collaborative projects in DH see, for example, Deegan, Neuroth.

⁶ www.monasterium.net.

objects. It is possible that an archivist with a special interest writes one abstract and a diplomatist or philologist contributes another separate view of the same charter.⁷ This concept of “parallel versions”⁸ will be relevant later on in this paper.

3 Versioning in software development

In software development, versioning is implemented by version control systems (VCSs), first deployed in the 1970s.⁹ Complex software programs consist of millions of lines of code, which are intended to function smoothly together. Independent of programming style, software development is an iterative process: the developer writes the code, tests it, rewrites and enhances it, and then the cycle starts again with the testing phase. Version control software tracks the iterations or changes made to the files at each step of this iterative process providing access to each version of the file. Each version has a timestamp and an author. At any time, it is possible to identify who changed what and when. All changes, accidental or not, can be reverted, and old statuses of files can be recovered. In a collaborative software development setting, people have the possibility to work simultaneously on the same code file(s). In this case a VCS coordinates possible conflicts within the code. The rationale of these systems is the avoidance of data loss (Baerisch 1–9).

Roughly grouped, there are two system architectures used for VCSs. On the one hand, there is the so-called centralized approach, which has been the standard for version control for many years. A code repository on a single server contains all the versioned files and clients receive the code from that central system. A well-known representative is *Apache Subversion*.¹⁰ On the other hand, there are distributed VCSs, such as *Git*, *Mercurial*, or *Bazaar*.¹¹ Distributed systems allow the clients to have a full copy of all files in the repository on their local machines, with the advantage that in the case of a server crash, every client can restore the data (Chacon and Straub).

All VCSs share similar concepts and common functionalities. One of these is that they provide repositories on one or more servers. These repositories contain all code files of a software project, which are administrated and versioned by the VCSs. Every user can have an up-to-date working copy of the dataset, which is the current version. The server administrates the repository and coordinates the versions from the clients and keeps all the versioning information.

Anyone who wants to work on the files must first check them out from the project’s code repository. This means the user gets a copy of the files and can choose if she

⁷ The importance of this feature is debated by Georg Vogeler: Vogeler 74.

⁸ “Parallel versions” in markup are discussed, e.g. by Buzzetti or by Vasold, as “Informationsräume.”

⁹ See i.g. Source Code Control System: sccs.sourceforge.net.

¹⁰ subversion.apache.org.

¹¹ github.com; www.mercurial-scm.org; bazaar.canonical.com/en/.

wants the latest or an earlier version of the code. In any case, the user can start to edit the local copy. If the changes are to be brought under version control again, the user sends a *commit* request to the VCS. The new code is then compared with the checked-out-code and the differences are saved, or like with Git, a snapshot of the files is recorded with a reference to its preceding version. This new code can be fetched from the repository by all other users. Hence, all users stay in sync with their data and get the latest updates. The update process is similar to the commit: the local project copy is compared with the current repository project and differences are merged into the local version.

When different users are simultaneously working on the same file, there is the potential (even the inevitability) that their changes will come into conflict.¹² In the case of multiple users working on the same lines of code and with concurrent changes being made on the same file as a result, the VCS does not merge these changes automatically on update. It reports the conflicting parts of the code and the user is asked how to integrate the relevant sections of code in order to solve the conflict. The changes to the local file copy will not be reflected in the repository until the user merges the conflicting parts in the file (Baerisch 9–15).

Branching in the development process supports experimentation with new ideas, such as when testing different feature implementations. The advantage of a *branch* is that changes in one branch do not affect other branches in the repository. A branch in this context means that at a certain point in the development two parallel versions of code are developed separately. In the introduction to the VCS called *Source Code Control System*, Eric Allman in 1980 (Nyman 73) notes “Creating a branch ‘forks off’ a version of the program.” By this process a copy of a program is made, which the current software development community a *fork*, though the differences between branches and forks are not so clear in the literature (compare Nyman). A fork, according to Robles and González-Barahona (3), happens when “a part of a development community (or a third party not related to the project) starts a completely independent line of development based on the source code basis of the project.” Furthermore, they argue that for something to be called a fork, there should be the following conditions: a new project name; a branch of the software; a parallel infrastructure; and a new developer community (Robles and González-Barahona 3). However, there are also more general definitions, as Nyman found in his studies and interviews:

Indeed, in addition to being used by some as a synonym with branch, fork may also now be seen used among developers to indicate to reusing [sic] existing code in the creation of a program that may target a significantly

¹² Although most modern VCS support collaborative editing, in more restrictive types of VCSs, this is not possible because only one user has editing rights to a file at the same time.

different user group than the original developed by hackers with no affiliation with the original project (Nyman 132).

To give a better understanding of versions, branches and forks in VCSs, the following figures summarise the main concepts and their consequent structures: Versions saved in a VCS form a line of versions (fig. 1). Versions occur in a sequence of time and are based on past versions.



Figure 1: Line of versions. In grey the most recent version.

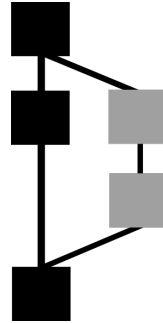


Figure 2: Temporary Branch in grey merged with the main branch.

A branch shapes the sequence of versions as a tree. Versions are developed in parallel (fig. 2). There is no definitive way to deal with branches (compare Nyman) but in general these are of temporary nature – either merged with the main branch (fig. 2) or deleted.

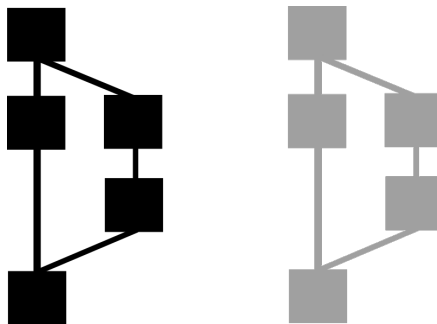


Figure 3: Original in black and fork in grey.

Forking can have different meanings within VCSs. In the *Git* model, forking belongs to the normal developer workflow when contributing to other open software projects (Chacon and Straub 123). A fork is thus a parallel project. The code base is copied with its history and developed in different ways independently from each other (fig. 3).

Finally, version control is not only used in software development, but also in the context of web content versioning. For example, *Wikipedia* uses a VCS to manage all user contributions and to display this information to them in the form of a history page.

4 Versioning in Wikipedia

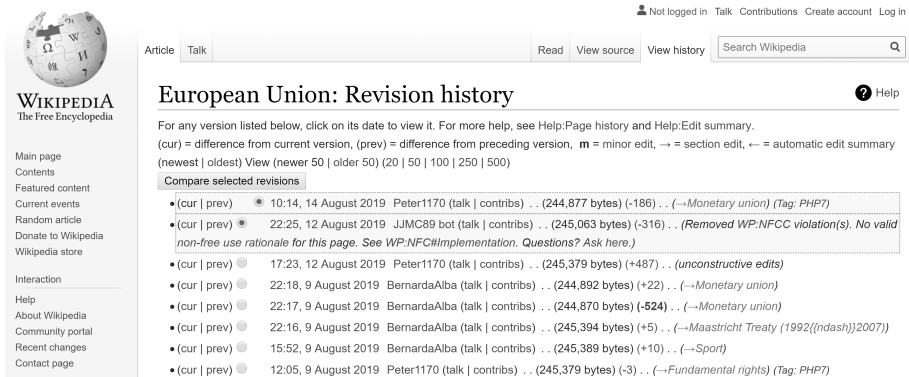
Wikipedia is one of the most visited web sites and popular collaborative editing platforms, which has been called “history’s biggest experiment in collaborative knowledge” (Poe). *Wikipedia* entries are created and updated every few seconds¹³, clearly distinguishing it from paper-based encyclopaedias. Additionally, it operates in more than 280 languages. Most of the lemmas explained on the platform exist in different languages not just as translations, but with its own original content for each language, no matter in what version. Consequently, the content may differ significantly: in some languages a concept is described in a well-researched way, whereas others have more rudimentary explanations, which reflect the *Wiki* philosophy of, and approach to, free content: “Wikipedia has no firm rules. Wikipedia has policies and guidelines, but they are not carved in stone” (Wikipedia contributors, “Five pillars”).

Wikipedia is largely written by interested laypeople, especially if the topic of the article is of general interest such as articles on pop-stars, movies, etc. There are no restrictions on contributors or content, and this fact can lead to a lively discussion process, and even to *edit wars*, when differences in perspective cannot be resolved.¹⁴ *Wikipedia* is aware of this problem and therefore reserves the right to close articles for editing. Before doing so, those involved are called upon to observe *Wikipedia etiquette*. *Wikipedia* is aware that articles are sometimes of poor quality (Wikipedia contributors, “Reliability”) and may contain false information, present only one single point of view, and the coverage of subjects might be heavily unbalanced. This happens when editors follow a special interest and ignore the requirement for an impartial and comprehensive perspective. However, *Wikipedia* believes in “its self-healing effects,” and interprets the aphorism of “Given enough eyeballs all bugs are shallow” (Raymond 30) as representing the way in which all content will improve over time

¹³ On the 26th March 2017, the English *Wikipedia* had 5,368,820 content articles and 41,786,208 pages in total that were generated by 881,544,043 edits. There are 30,550,216 registered users with 1,268 administrators (Wikipedia contributors, “About”).

¹⁴ On edit wars, see: Kallass 305–309.

given enough edits (Wikipedia contributors, “About”). Nevertheless, it is frequently necessary to reverse malicious edits, and for that purpose a VCS saves versions of all edits made to an article.



The screenshot shows the Wikipedia revision history page for the article "European Union". At the top, there is a navigation bar with "Article" and "Talk" tabs, and a search box. Below the navigation bar, the title "European Union: Revision history" is displayed. A help icon is visible on the right. The main content area contains a list of revisions, each with a radio button for selection, a timestamp, the editor's name, and details about the edit (size, deletions, insertions, and comments). The current version is selected with a bolded "m" in the comment.

Not logged in | Talk | Contributions | Create account | Log in

Article | Talk | Read | View source | View history | Search Wikipedia

European Union: Revision history

For more help, see [Help:Page history](#) and [Help:Edit summary](#).
 (cur) = difference from current version, (prev) = difference from preceding version, m = minor edit, → = section edit, ← = automatic edit summary
 (newest | oldest) View (newer 50 | older 50) (20 | 50 | 100 | 250 | 500)

Compare selected revisions

- (cur | prev) 10:14, 14 August 2019 Peter1170 (talk | contribs) . . (244,877 bytes) (-186) . . (→Monetary union) (Tag: PHPT)
- (cur | prev) 22:25, 12 August 2019 JJMC89 bot (talk | contribs) . . (245,063 bytes) (-316) . . (Removed WP:NFC violation(s). No valid non-free use rationale for this page. See WP:NFC#Implementation. Questions? Ask here.)
- (cur | prev) 17:23, 12 August 2019 Peter1170 (talk | contribs) . . (245,379 bytes) (+487) . . (unconstructive edits)
- (cur | prev) 22:18, 9 August 2019 BernardaAlba (talk | contribs) . . (244,892 bytes) (+22) . . (→Monetary union)
- (cur | prev) 22:17, 9 August 2019 BernardaAlba (talk | contribs) . . (244,870 bytes) (-524) . . (→Monetary union)
- (cur | prev) 22:16, 9 August 2019 BernardaAlba (talk | contribs) . . (245,394 bytes) (+5) . . (→Maastricht Treaty (1992{{ndash}}2007))
- (cur | prev) 15:52, 9 August 2019 BernardaAlba (talk | contribs) . . (245,389 bytes) (+10) . . (→Sport)
- (cur | prev) 12:05, 9 August 2019 Peter1170 (talk | contribs) . . (245,379 bytes) (-3) . . (→Fundamental rights) (Tag: PHPT)

Figure 4: Revision history page of Wikipedia.

All text versions of a specific *Wikipedia* page are accessible to the users via a history. The revision history page (fig. 4) lists all changes of any editable *Wikipedia* page. The edits are listed from newest to oldest, and for each edit the following details are listed: time and date of the edit, username of the editor or IP address for not registered users, the size of the file, the sum of deletions and insertions and optionally an edit comment. In the case of long edit histories, the user can navigate the versions by year and month. There are links to the most recent and oldest edits and the previous and next page in the history. Each listed version is shown in comparison to the current version by clicking “(cur)”, while by clicking the “(prev)” link next to each version, the previous version is shown on a separate page. Any two versions on the page can be compared by clicking the radio buttons in the lines of the selected versions. *Wikipedia* distinguishes between minor and major changes: a bolded “m” in the revision history signals that only minor changes have been made to the preceding version. *Wikipedia* defines minor and major edits as follows:

A check to the minor edit box signifies that only superficial differences exist between the current and previous versions. Examples include typographical corrections, formatting and presentational changes, and rearrangements of text without modification of its content. A minor edit is one that the editor believes requires no review and could never be the subject of a dispute. An edit of this kind is marked in its page’s revision history with a lower case, bolded “m” character (**m**).

By contrast, a major edit is one that should be reviewed for its acceptability by all the editors concerned. Any change that affects the meaning of an article is not minor, even if it concerns a single word; for example, the addition or removal of "not" is not a minor edit (Wikipedia contributors, "Minor edit").

These categories allow for a weighting between more and less substantial changes in a version history. The most interesting changes are those emerging from a debate in which an editor succeeds in imposing her will. This usually happens because it is the strategy of *Wikipedia* to develop one article per lemma and conflicting parts arising from different perceptions have to be consented to and merged. It is the same with conflicting code in software development unless you fork. This is not the case in research praxis, where a multiplicity of opinions and perspectives is important.

5 Versioning scholarly positions

As mentioned above, the collaboratively generated content can be overwritten, corrected, or continued; however, in all instances just the most recent version is displayed. This happens when the versioning strategy is realized as a one-to-one relationship of versions. The version represents one possible intellectual position. But, what if we want to represent different research positions or interpretations of the same edited object? Should there always be a privilege for the most recent version? In a scholarly discourse this is not acceptable. However, if we want to use VCSs to make scholarly positions transparent and documented, it is necessary to adopt a versioning strategy, which enables multiple perspectives as well as dissent. Meister defines key requirements for a collaborative approach to textual markup which can be effective within laboratories as well. In a collaborative setting, it should be possible to relate one source object with more than one describing instance. Meister states:

Collaborative markup must be based on a one-to-many relationship model in which one object text is related to n associated markup instances [...] every markup instance should be preserved as unique data set, with a pointing relation to the source document. [...] All original source documents and all the markup instances related to these must be handled by an integrated document management system [...] (120)

VCS could handle document management. As mentioned above, VCSs provide the ability to develop and manage software code in parallel. Especially forking, in the sense of spin-off and new development, offers new possibilities for the scholarly discourse. The "right to fork" (Nyman 1), which is so important for the open source software community, can be adopted for scholarly purposes. With a fork, you have the possibility to develop in a new direction something independent from the predecessor

or origin. The crucial difference from just copying the research object is that, in the case of a fork, the reference to the origin that is being copied remains. This means that VCSs could easily manage different views on research objects.

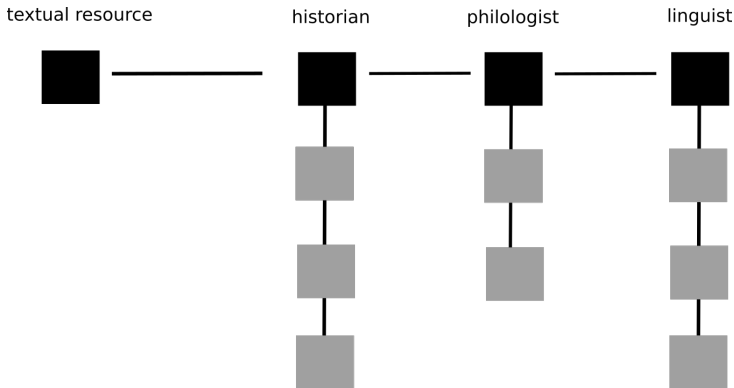


Figure 5: Forks as parallel positions in black, individual versions in grey.

In this way, as illustrated in figure 5, a textual resource can be forked and as such analysed by, e.g., a historian. A philologist, on the other hand, could set up a fork with his or her interest and questions, and similarly other researchers with different interests and points of view could do the same. This means that by forking through a VCS, scientific discourse is trackable, thus guaranteeing the plurality of interpretations.

In a VCS versions are administrated entities without further semantic meaning. Each change is a new version in the project under version control. Users can compare versions and see what has been changed; afterwards, they may evaluate and interpret the changes. In general usage, as a term, version is versatile and not very precisely defined (see Nury in this book). In some contexts correction of spelling may not be considered a reason to form a new version, but for others they are—and VCSs agree with the latter, since every saved change creates a new version.

The concept of a fork (respectively branch) helps to distinguish semantically and technically from versions. Forks specify parallel versions. Parallel versions have the same original versions, but do not overwrite each other every time they are changed. However, similar to versions, the meaning of each fork and its purpose can vary greatly and we do not know its specific purpose. Only in the comparison between the fork and its origin or with other forks does the meaning become clear. Would it not be desirable to determine the type of change, to know the reason for a parallel development? In concrete terms, would it not be interesting for the scientific discourse

to relate the different versions in order to make the significance of each version and of their contribution to the full collaborative process clear?

An approach in software development that takes the semantics of changes into account is *Semantic Versioning*, which suggests a version notation of software releases in order to more easily determine compatibilities with different software packages. Tom Preston-Werner's prescription is as follows:

Given a version number MAJOR.MINOR.PATCH, increment the: MAJOR version when you make incompatible API changes, MINOR version when you add functionality in a backwards-compatible manner, and PATCH version when you make backwards-compatible bug fixes.

Following his criteria, the notation is meaningful for all developers and users. He differentiates three types of versions. With the classical VCS, we can differentiate two types: a version and a parallel version. However, if we want versions to be meaningful, they must be differentiated in relation to other versions, such as: "this is an alternative point of view," "this is a contraposition," "this is an addition," or "just revision and not really a new version."

In the following, I will present well-known data models in the field of DH, which can extend the traditional versioning model. They promise to document dynamic objects and their changes over time. Each of the three examples comes from a different domain, and each has its own concept and way of dealing with change, as well as processes that produce versions: FRBRoo, OAI-ORE and PROV-O.

6 Three examples for modelling versions

6.1 Versioning with FRBRoo

The knowledge of libraries and museums is represented in the FRBRoo ontology, which was published in 2009 as an extension of CIDOC CRM¹⁵ and FRBR.¹⁶ FRBRoo is a very large ontology; however in this section, just a few aspects are considered because of their relevance to the modelling of versions. Let us first look at the concepts of *Work* and *Expression*.¹⁷ The notion of *Work* is described as:

A distinct intellectual or artistic creation. A *work* is an abstract entity; there is no single material object one can point to as the *work*. We recognize the *work* through individual realizations or *expressions* of the *work*, but the *work*

¹⁵ International Council for Museums – International Committee on Documentation. Ifla.com

¹⁶ Functional Requirements for Bibliographic Records. Ifla.com.

¹⁷ In FRBR and FRBRoo Manifestation and Item do not include web resources. An adaption for digital documents is suggested by Albertsen and Van Nuys.

itself exists only in the commonality of content between and among the various *expressions* of the *work* (IFLA 54).

Expression is already mentioned in the above definition as a realisation of a *Work*. More fully:

Inasmuch as the form of *expression* is an inherent characteristic of the *expression*, any change in form (e.g., from alpha-numeric notation to spoken word) results in a new *expression*. Similarly, changes in the intellectual conventions or instruments that are employed to express a *work* (e.g., translation from one language to another) result in the production of a new *expression*. If a text is revised or modified, the resulting *expression* is considered to be a new *expression*. Minor changes, such as corrections of spelling and punctuation, etc., may be considered as variations within the same *expression* (IFLA 55).

This definition comes from a real world problem for librarians: obviously, there are differences from the world of software developers and the use of VCSs. What in a VCS is a version, in FRBROO is either a variation within the same expression or a new expression. Not every change generates a new version as expression: minor changes would not have an impact on the expression.

In FRBROO, dynamic aspects of objects can be represented by *events*. Events bear the chance to identify similarities of, or differences between, objects. By describing the event of e.g. creation, which is situated in time and space and involves actors, FRBROO expression becomes a comparable or exchangeable information (fig. 6). There are concepts for events that deal especially with the creation, publication and production processes.¹⁸

Applied to the process of versioning, all versions are expressions and would be generated by events, because it is the activity *commit* that brings a version into existence. Furthermore, the version that is generated by the creation event could be modelled with a timestamp and a responsible person. Then in a collaborative research setting new versions could be created by modifying the former version. In this case, a type like “Revision” or “Adaption” can be added. In order to describe the relationships between versions’ properties, such as “consists of,” “influenced by,” “has modified,” “is composed of,” “refers to,” “is logical successor of,” “is derivative of,” or “is based on” could be used. FRBROO and CIDOC CRM allow abundant possibilities to describe version relationships between cultural heritage objects, especially because both provide more than 230 different relationships between entities.

¹⁸ Besides these events, there are terms for “Recording Event,” “Performance,” and “Reproduction Event” (IFLA).

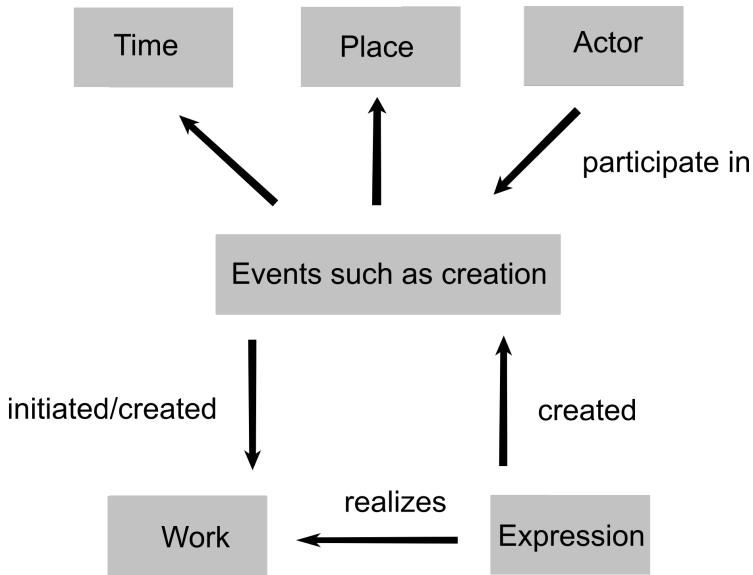


Figure 6: Temporal entities modelling intellectual processes (simplified).

6.2 Versioning with the Open Archive Initiative

The *Open Archive Initiative* (OAI) has its roots in the development of e-print repositories, so-called archives. Version 1.0 of the *Object Reuse and Exchange (ORE)* data model specification was released in 2008 and no further version has been released yet.

The strength of the *OAI-ORE* is in particular in describing and exchanging web resources. It aims to provide concepts to describe their constituents and boundaries. Furthermore, it is possible to easily compound information objects to a logical whole (Lagoze and Van de Sompel, “Compound”). Usually web resources are *compound information objects*. They can be located on a single web server or be distributed anywhere. However, these resources are somehow related to each other forming a logical entity and *OAI-ORE* is used to encapsulate and document their relationships. An example can give an idea of what is meant by this: an entry on a collaboratory can be described as a combination of all involved software, interlinked HTML pages, facsimiles and text documents as well as all versions of software, HTML pages, facsimiles and texts. All these resources can be convened through one information object, whose components and relationships are described together.

Compared to *FRBRoo* or *CIDOC CRM*, the *OAI-ORE* abstract data model is lightweight. The model differentiates four classes, representing the core entities of interest:

firstly, the class *Aggregation*, which is a pure conceptual construct that aggregates the set of interesting resources. Van de Sompel and Lagoze call it a containment node (“Archive”), i.e. a fictive, empty node with a HTTP URI that binds the resources to the compound object and as such defines its boundaries as well (Open Archive Initiative).

An Aggregation is a compound of resources. One of these resources is an instance of the class *Aggregated Resource*. All Aggregated Resources, in turn, constitute the Aggregation. The Aggregated Resources can differ in media type and format. Each Aggregated Resource has its own URI. All resources belonging to one compound information object are listed in a *Resource Map* (Open Archive Initiative).

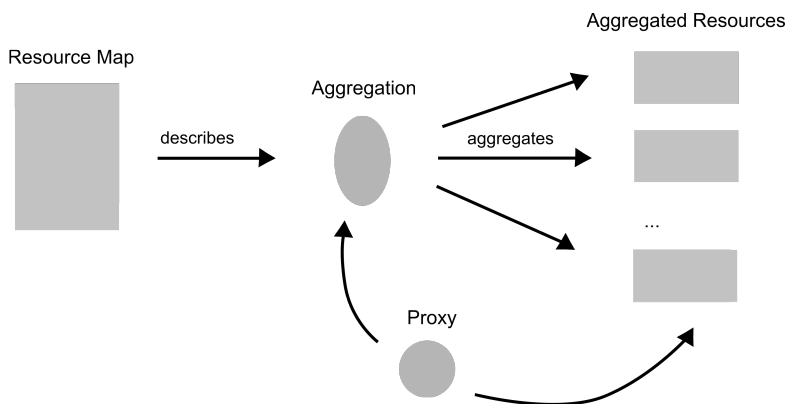


Figure 7: Illustrating four classes of the OAI-ORE model.

The *Resource Map* class contains all the information for machine consumption. It is an encoded description of the compound information object, which has a separate URI. The Resource Map lists the Aggregation and all Aggregated Resources, relates the single resources to each other, and assigns properties. As part of the obligatory description, it contains metadata of the author as well as a date-timestamp. The Resource Map provides the description of exactly one Aggregation, which forms one of various possible representations of the combined resources (Open Archive Initiative).

The *Proxy* class provides a method for ordering the resources within compound objects. The Aggregated Resources do not have an inherent order for serializing the data: in the *OAI-ORE* model this is done with the help of *Proxies*. The relationships between Proxies define the order of the resources and the Aggregated Resources, which they represent, stay neutral, without further context information. A Proxy has its own URI name, which is listed together with all the other information in the Resource Map (Open Archive Initiative).

In a collaboratory, research tracked with OAI-ORE would enable the sharing and reuse of resources. Every user creates her own set of resources: for example, a source (images) and corresponding research results (text documents). Different versions of text, like intermediate results or partial results, and images would be separately addressable resources, which can be sorted chronologically by proxies. When another collaborator continues working on a version (resource), it is aggregated and is therefore part of another set. When work continues on this aggregated version, a new resource is created - represented by a proxy and linked to the previous version. This means that any number of descriptions can be created in a collaborative research environment. References to other resources (versions) can be created. Even a whole set can be aggregated again as a resource and thus become an object of further consideration. This fosters a free give and take that “provide(s) the foundation for advanced scholarly communication systems that allow the flexible reuse and refactoring of rich scholarly artifacts and their components”(ORE Specification). When it comes to the exact description of relationships, OAI-ORE does not offer many possibilities. However, the standard is open to integration with other vocabularies that provide useful terms in their domains (Open Archive Initiative), a process that would be necessary for an adequate description of the versions. The *OAI-ORE* data model recommends the use of *Dublin Core Elements*, *Dublin Core Terms*, *Friend of a Friend* terms, *RDF* terms, and *RDF Schema* terms, as well as terms from *PROV-O*, which is presented in the next section.

6.3 Versioning with the Provenance Ontology

The *Provenance Ontology* (PROV-O) is a domain-agnostic ontology, recommended by the *World Wide Web Consortium* (W3C)¹⁹ in 2013. Provenance is defined as “a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of data or a thing” (W3C). In any case, it provides a better understanding of contexts. Provenance information intends to increase the trustworthiness, reliability and quality of the described resources (W3C). The main purpose of the *PROV* is to provide a standardised way for representing and exchanging domain-independent provenance information. In its core, the *PROV* data model is about *entities*, *activities*, and *agents*, who are interconnected by relationships (see fig. 8).

Entities are all things of interest about which we ask provenance information. These can be physical items such as a printed book or any other artefact, but also abstract ideas and digital objects as web pages and files. The definition is: “An entity is a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities

¹⁹ Founded in 1994, the W3C develops standards for the World Wide Web: www.w3.org.

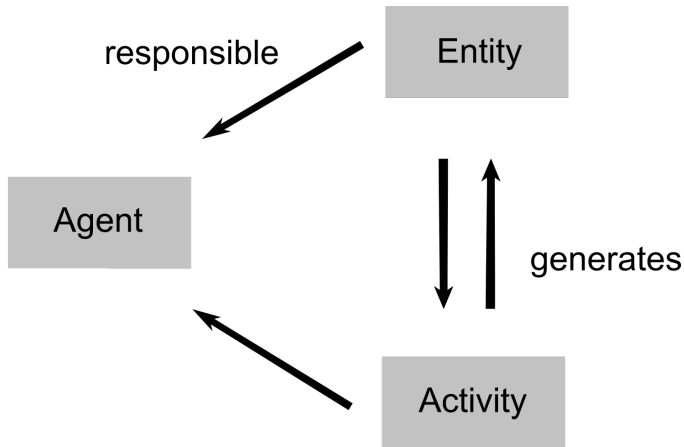


Figure 8: Core concepts of PROV-DM.

may be real or imaginary” (W3C). Instead, an *Activity* is something that occurs to entities, interacts with entities over a period of time or produces new entities. “It may include consuming, processing, transforming, modifying, relocating, or generating entities” (W3C).

An entity which is based on a pre-existing object is called a *Derivation*. A special case of derivation is the concept of *Revision*. A Revision is an entity that implicitly contains substantial content of the original entity: it “is a derivation for which the resulting entity is a revised version of some original” (W3C).

Provenance is described from different points of view, with different foci on agents or entities and their activities. Hence, the ontology considers concepts that are about distinguishing the same thing under different conditions and perspectives. When there is more than one provenance description for the same thing, these description entities can be related through the concepts *Specialization* and *Alternate*.

Right after the last release of the PROV-DM in 2013, De Nies et al. wrote a web service to convert processes in a VCS into PROV. In order to make provenance information within VCS interoperable and exchangeable, Arndt et al. continue and extend this approach and suggest a semantic representation of a Git commit, which as RDF-graph can be queried via a public API. They mapped a commit as an activity, associated with an author and a committer and with a start and an end time. Since they were working to version RDF-graphs, their entities were named graphs in a file linked to the commit via *wasGeneratedBy* and which attributed its origin as *specializationOf*. They further enriched the functionalities of Git with metadata; their

approach, however, does not describe the semantic meaning of the individual versions. PROV provides the concepts *Revision*, *Derivation* or *Alternate* exactly for this purpose, to avoid unspecific versions.

7 Conclusion

Over the last 15 years, collaboratories have brought together the efforts of many contributors to collaboratively conduct and share research. However, in order to make collaboratively created research output and its creation process transparent, all edits have to be managed and made citable and accessible. A minimal solution would be that collaboratories provide a version history to their users. Many of the platforms already have this kind of information because their databases are under version control and it is important to pass this information on to the users in order to verify and replicate the various contributions. It must be possible to document the issues we communicate on with a standardized approach and in such a way that they are made accessible for all.

A challenge still unresolved is the need to work together while allowing contributors to follow different interests and investigate different research questions. If we wish to establish a scholarly discourse in collaboratories, we have to create the space for multiple opinions. Yet we cannot apply *Wikipedia's* principle of *Linus' law* i.e. all viewpoints must be merged in order to reach a consensus. I would argue that the *Wikipedia* principle is not great for research platforms and a better approach should be in concordance with Raymond's statement: *Given enough eyeballs any consensus is shallow*. In other words, dissent is important in academia and research platforms must be able to accommodate it.

In this paper it was proposed to implement versioning in collaboratories as extended version history. By increasing the significance of version control, research transparency and critical discussion could be improved. The current model of versioning has to be extended in this direction, and the ontological models presented in this article indicate potential ways that this can happen by connecting versions through specific relationships. All of these models draw our focus to the relationships between entities, and by doing so they force us to pay more attention to the events occurring to our information resources and their contributors.

Bibliography

- ABO. Annotated books online*, edited by Mathijs Baaijens, et al. www.annotatedbooksonline.com. Accessed 31 March 2017.
- Albertsen, Ketil, and Van Nuys, Carol. "Paradigma FRBR and Digital Documents." *Functional*

- Requirements for Bibliographic Records (FRBR). Hype or Cure-All?* edited by Patrick Le Boeuf, Haworth Information Press, 2005, pp. 125–50.
- Allman, Eric. “An Introduction to the Source Code Control System”, sccs.sourceforge.net/~man/sccs.me.html. Accessed 13 Jan 2019.
- Arndt Natanael et al. “Exploring the Evolution and Provenance of Git Versioned RDF Data.” *Joint proceedings of the 3rd Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW 2017) and the 4th Workshop on Linked Data Quality (LDQ 2017) co-located with 14th European Semantic Web Conference (ESWC 2017), CEUR Workshop Proceedings*, vol. 1824, edited by Jeremy Debattista, Jürgen Umbrich, and Javier D. Fernández, 2017. ceur-ws.org/Vol-1824/mepdaw_paper_2.pdf. Accessed 3 June 2019.
- Baerisch, Stefan. “Versionskontrollsysteme in der Softwareentwicklung.” *IZ Arbeitsberichte*, edited by Informationszentrum Sozialwissenschaften der Arbeitsgemeinschaft Sozialwissenschaftlicher Institute e.V. (ASI), vol. 36, 2005, www.gesis.org/fileadmin/upload/-forschung/publikationen/gesis_reihen/iz_arbeitsberichte/ab_36.pdf. Accessed 26 Aug 2019.
- Becker, Hans-Georg. “FRBR, Serials and CIDOC CRM – Modellierung von fortlaufenden Sammelwerken unter Verwendung von FRBRoo.” *(Open) Linked Data in Bibliotheken*, edited by Patrick Danowski and Adrian Pohl, De Gruyter, 2013, pp. 64–96.
- Blischak, J.D. et al. “A Quick Introduction to Version Control with Git and GitHub.” *PLoS Comput Biol*, vol. 12, no. 1, 2016: e1004668. DOI: 10.1371/journal.pcbi.1004668.
- Buzzetti, Dino. “Digital Representation and the Text Model.” *New Literary History*, vol. 33, 2002, pp. 61–88.
- Carusi, Annamaria and Reimer, Torsten. *Virtual Research Environment Collaborative Landscape Study*. JISC, 2010.
- Chacon, Scott, and Straub, Ben. *Pro Git*. git-scm.com/book/en/v2/Getting-Started-About-Version-Control. Accessed 3 March 2017.
- CIDOC. *Definition of the CIDOC Conceptual Reference Model*, version 6.2 (May 2015). Edited by Patrick Le Boeuf et al. www.cidoc-crm.org/sites/default/files/cidoc_crm_version_6.2.pdf. Accessed 3 March 2017.
- Danowski, Patrick and Pohl, Adrian, editors. *(Open) Linked Data in Bibliotheken*. De Gruyter, 2013.
- Deegan, et al. *Collaborative Research in the Digital Humanities: a Volume in Honor of Harold Short, on the Occasion of His 65th Birthday and His Retirement, September 2010*. Ashgate, 2012.
- De Nies, Tom, et al. “Git2PROV: Exposing Version Control System Content as W3C PROV.” *Proceedings of the ISWC 2013 Posters & Demonstrations Track a track within the 12th International Semantic Web Conference (ISWC 2013), CEUR Workshop Proceedings*, vol. 1035, edited by Eva Blomqvist, and Tudor Groza, 2013. ceur-ws.org/Vol-1035/iswc2013_demo_32.pdf.
- IFLA. *Definition of FRBRoo. A Conceptual Model for Bibliographic Information in Object-Oriented Formalism*. edited by Bekiari, Chryssoula et al. IFLA, 2016. www.ifla.org/files/assets/cataloguing/FRBRoo/frbroo_v_2.4.pdf. Accessed 3 March 2017.
- Flanders, Julia. “Collaboration and Dissent: Challenges of Collaborative Standards for Digital Humanities.” *Collaborative Research in the Digital Humanities*, edited by Marilyn Deegan

- and Willard McCarty, Ashgate, 2012, pp. 67–80.
- ICARUS. *Monasterium.net*. www.monasterium.net. Accessed 31 March 2017.
- Kallass, Kerstin. *Schreiben in der Wikipedia. Prozesse und Produkte gemeinschaftlicher Textgenese*. Springer, 2013.
- Lagoze, Carl, and Van de Sompel, Herbert. “The Open Archives Initiative: Building a low-barrier interoperability framework”. OAI, 2001, www.openarchives.org/documents/jcdl2001-oai.pdf. Accessed 31 March 2017.
- . “Compound Information Objects: The OAI-ORE Perspective” OAI, 2007, www.openarchives.org/ore/documents/CompoundObjects-200705.html. Accessed 24 March 2017.
- Le Boeuf, Patrick “A Strange Model Named FRBRoo.” *Cataloging & Classification Quarterly*, vol. 50, no. 5–7, 2012, DOI: 10.1080/01639374.2012.679222, pp. 422–38.
- . “FRBR: Hype or Cure-All? Introduction.” *Functional Requirements for Bibliographic Records (FRBR). Hype or Cure-All?* edited by Patrick Le Boeuf, Haworth Information Press, 2005, pp. 1–13.
- Le Boeuf, Patrick, and David Miller. “Such Stuff as Dreams Are Made On: How Does FRBR Fit Performing Arts?” *Functional Requirements for Bibliographic Records (FRBR). Hype or Cure-All?* Edited by Patrick Le Boeuf, Haworth Information Press, 2005, pp. 151–178.
- Meister, Jan-Christoph. “Crowd Sourcing ‘True Meaning’: A Collaborative Markup Approach to Textual Interpretation” *Collaborative Research in the Digital Humanities*, edited by Marilyn Deegan, Willard McCarty, Ashgate, 2012, pp. 105–122.
- Neuroth, Heike, et al. *TextGrid: Von Der Community – Für Die Community: Eine Virtuelle Forschungsumgebung Für Die Geisteswissenschaften*. Verlag Werner Hülsbusch [Printversion] in Kooperation Mit Dem Universitätsverlag Göttingen [Onlineversion] 2015. DOI: doi.org/10.3249/webdoc-3947.
- Nyman, Linus. “Understanding Code Forking in Open Source Software. An Examination of code forking, its effect on open source software, and how it is viewed and practiced by developers.” Helsinki 2015, helda.helsinki.fi/bitstream/handle/10138/153135/287_978-952-232-275-3-1_v2.pdf?sequence=5&isAllowed=y.
- Open Archives Initiative. *ORE Specification – Abstract Data Model*, edited by Carl Lagoze, Herbert Van de Sompel, 2008. www.openarchives.org/ore/1.0/datamodel. Accessed 22 March 2017.
- Papyri.info*, edited by the Duke Collaboratory for Classics Computing & the Institute for the Study of the Ancient World, papyri.info. Accessed 3 March 2017.
- Poe, Marshall. “The Hive.” *The Atlantic*, September 2006, www.theatlantic.com/magazine/archive/2006/09/the-hive/305118. Accessed 27 March 2017.
- Preston-Werner Tom. “Semantic Versioning 2.0.0.” semver.org/. Accessed 13 Jan 2019.
- Raymond, Eric. *The Cathedral & the Bazaar*. Musings on Linux and Open Source by an Accidental Revolutionary. O’Reilly, 2001.
- Robles, Gregorio and Gonzalez-Barahona, Jesus M. “A Comprehensive Study of Software Forks: Dates, Reasons and Outcomes.” *Open Source Systems: Long-Term Sustainability*, edited by I. Hammouda et al., OSS 2012. IFIP Advances in Information and Communication Technology, vol 37, Springer, 2012.
- Suda On Line: The Byzantine Lexicography*. Edited by Stoa Consortium, www.stoa.org/sol.

- Accessed 31 March 2017.
- Vasold, Gunter. "Progressive Editionen als multidimensionale Informationsräume." *Digital Diplomatics: the Computer as a Tool for the Diplomatist?* edited by Ambrosio, Antonella, et al., Böhlau, 2014, pp. 75–88.
- Visser, A.S.Q, and Calis, R.A. "Building a Digital Bookwheel Together: Annotated Books Online and the History of Early Modern Reading Practices." *Bibliothecae.it. Rivista di studi semestrale*, vol. 3, no. 1, 2014, pp. 63–80.
- Vogeler, Georg. "Das Verhältnis von Archiven und Diplomatik im Netz. Von der archivischen zur kollaborativen Erschließung." *Digitale Urkundenpräsentationen*, edited by Joachim Kemper and Georg Vogeler, Books on Demand, 2011, pp. 61–82.
- Wikipedia contributors. "About." *Wikipedia, The Free Encyclopedia*, 2017, en.wikipedia.org/w/index.php?title=Wikipedia:About&oldid=771132497. Accessed 19 March 2017.
- . "Five pillars." *Wikipedia, The Free Encyclopedia*, 2017, en.wikipedia.org/w/index.php?title=Wikipedia:Five_pillars&oldid=769798419. Accessed 11 March 2017.
- . "Minor edit." *Wikipedia, The Free Encyclopedia*, 2017, en.wikipedia.org/w/index.php?title=Help:Minor_edit&oldid=764768221. Accessed 26 March 2017.
- . "Reliability." *Wikipedia, The Free Encyclopedia*, 2017, en.wikipedia.org/w/index.php?title=Reliability_of_Wikipedia&oldid=772272370. Accessed 26 March 2017.
- W3C. *PROV-DM: The PROV Data Model*. edited by Luc Moreau, Paolo Missier, W3C, 2013. www.w3.org/TR/2013/REC-prov-dm-20130430. Accessed 18 March 2017.