



Version Management in a Distributed Infrastructure for Open Educational Resources

Nadine Schroeder

Abstract

One concern of Open Educational Resources (OER) is to establish infrastructures, such as repositories, where learning materials can be uploaded and exchanged. Various initiatives all over the world are currently investigating technical developments for finding and sharing OER in higher education. In this context, the consolidation of individual solutions in a distributed infrastructure must be considered. When creating and editing content, modifications and adjustments can result in new versions of a resource and further developments of other users can lead to derivatives. Managing versions in terms of tracking changes and learning about new versions available is not only an issue for OER repository development, but also for interaction and discoverability in a distributed infrastructure. Therefore, version management can be considered as an approach to potentially improve the reuse and revision OER. This contribution discusses use cases of OER in the context of version management and presents approaches to managing educational material in a distributed infrastructure resulting in a concept of version management for OER.

N. Schroeder (✉)

Department of Educational Sciences, Learning Lab, University of Duisburg-Essen, Essen, Germany

e-mail: nadine.schroeder2@uni-due.de

© The Author(s) 2023

D. Otto et al. (eds.), *Distributed Learning Ecosystems*,
https://doi.org/10.1007/978-3-658-38703-7_13

241

1 Introduction

A central issue of Open Educational Resources (OER) is to establish infrastructures, such as repositories, where learning materials can be uploaded and exchanged. In higher education, various initiatives are currently investigating technical developments for finding and sharing OER. In addition to single OER repositories, distributed infrastructures, in which individual solutions are brought together to make OER more discoverable, also matter (see Sect. 2). When designing infrastructures for OER, version management is one relevant topic, as new versions occur when materials are created, reused, or edited.

To manage versions, version control is one solution which is widely used in software development as a concept for working on source code collaboratively as well as storing versions, tracking changes, and copying content for further development. Version control systems like GitHub are openly accessible hosting platforms for software code and allow developing and providing open source code. This aspect also applies to openly licensed educational materials. Although OER cover different types of material and file formats than software code, version management functions can be transferred as use cases for OER (see Sect. 3). When creating and editing OER, modifications and adjustments can result in new versions of a resource. Therefore, displaying an overview of versions with details of their differences are helpful for better comprehensibility. Especially when content is created collaboratively, the advantages of version management become clear. There is no overlap of changes and revisions can be transparently assigned to people so they can prove their involvement in the creation of the content. Furthermore, other users can modify content and develop it as their own resource. Thus, implementing the idea of OER, that materials can be used by others. In addition, the original author can receive recognition for a high-quality resource based on the number of reuses. Therefore, version management can be considered as an approach to potentially improving the reuse and revision of OER. Still, transferring the versioning of OER onto platforms used for software development, such as GitHub, faces barriers. Users of OER without a technical background might find the procedures and application of these functionalities challenging and discouraging so that user-friendly interfaces are needed.

A further challenge concerns a distributed infrastructure for OER, where different aspects of connecting and referencing versions need to be considered to maintain consistent version information (see Sect. 4). The distributed and modifiable nature of OER results in users creating and sharing materials on different platforms so that the issue of duplicates needs to be considered. Besides, with a

possibly high number of versions for one resource, search functionalities of a distributed infrastructure need to consider the consolidation of versions.

Consequently, this contribution aims at discussing OER in the context of version management and presenting approaches to managing OER in a distributed infrastructure, resulting in a concept of version management for OER. After describing the theoretical background of technical OER infrastructure and distributed version control (see Sect. 2), functions of version management are linked to possible use cases for OER (see Sect. 3). Finally, a concept for managing versions of OER in a distributed infrastructure, addressing issues such as metadata and persistent identifiers, tracking changes, further developments, as well as availability of new versions will be presented (see Sect. 4).

2 Theoretical Background

2.1 Technical Infrastructures of OER

The topic of OER is part of theoretical discussions and practical developments of technical infrastructures that enable educators and learners to find and share educational materials (Clements et al., 2015; Heck et al., 2020; Santos-Hermosa et al., 2017). OER repositories for storing OER are predominantly created by and for higher education institutions in Europe as well as the USA and are mainly designed for multidisciplinary educational resources (Santos-Hermosa et al., 2017). As the structure of educational systems is decentralized in most countries, higher education institutions have already established infrastructures to store OER. In Germany, institutional repositories exist at individual universities and in some federal states. Some repositories that enable the provision of OER for higher education teachers are already operating at certain institutions, others are still under development. To bring these individual solutions together, a common metadata standard is being discussed (see chapter Menzel). A distributed learning ecosystem can be seen as one solution for connecting OER services and improving aggregation of content and resources (see chapter Otto & Kerres). The idea of a distributed infrastructure for OER is based on a concept where different repositories and platforms containing OER are connected to a Core Hub through which the exchange of metadata takes place (Kerres et al., 2019). This cross-linked system aims at supporting findability and accessibility of OER.

Beyond the availability and accessibility of materials, the collaborative creation and use of teaching and learning materials are also part of the OER concept. This is reflected in the four elements *Search*, *Share*, *Reuse*, and *Collaborate*,

which should be supported by an infrastructure (Atenas & Havemann, 2014). However, current repositories are mainly designed for storing and finding OER rather than fostering collaboration and social interaction between users, even though collaborative instruments enable users to participate in repositories and to develop OER together (Clements et al., 2015). Alongside collaborative features an active user community is needed to enhance the quality of OER and repositories (Zervas et al., 2014).

2.2 Version Management and OER

A version management system is a system used to record changes to documents or files (Franzetti, 2019). For each version of a file, information such as author name, time of change, and change notes is stored as the current status. In this way, changes can be tracked and older versions can be accessed or restored (Vijayakumaran, 2019). The advantage of distributed version control, such as Git, is that several users can asynchronously change the same version of a file, as the local changes are synchronised to a new version on the central server (Zolkifli et al., 2018). Particularly through the popular hosting platform GitHub, distributed version management has become publicly possible and simplifies collaborative work, thereby significantly supporting the open source movement. Software developers use version management to jointly create and edit code both internally and with external collaborators. Contributors to a project can propose changes that can be accepted or rejected by the maintainer and merged into the previous version. Changes can be tracked transparently and assigned to individual people.

In the context of *reusable learning objects*, version management was discussed (Brooks et al., 2003) and exemplarily realised for course content on a platform designed for creating, sharing, and reusing course content based on markdown (Salas, 2020). However, in this case, the course material was not published in an open repository, but within an access-restricted institutional platform. If content is not shared publicly, then licensing issues do not play a dominant role and collaboration is facilitated by the fact that authors highly likely know each other.

There are just a few use cases in the literature for the version control system GitHub as a collaborative learning environment where teachers and learners can interact within a course (Zagalsky et al., 2015). In another example, a research project showed that GitHub is used for the provision and storage of educational resources and that changes are mainly made by the project owners. However, other advantages of version management, such as copying and editing

of materials by external people, are only used to a small extent (Schroeder & Pfaender, 2020). To exploit the potential of version management for the use of OER, one possible solution may be to transfer the version management processes into a user-friendly interface that facilitates access for users without technical background as GitHub, seems to be challenging for them (Ovadia, 2019).

3 Adaptation of Version Management for OER

The idea of OER is related to the 5R-concept according to Wiley (2014). These user rights describe options of open licenses when dealing with learning and teaching materials: Access to materials and the permission to save materials and their files as a personal copy (*retain*) are the prerequisite for the other rights to use, edit, and share OER. When dealing with version management, the aspects of *reuse*, *revise*, and *remix* matter while creating and editing new versions of OER. Reuse is possible by integrating content without changing it, while revising can be done by removing, adding, or rearranging content before using it in one's own material. Remixing means combining and changing different OER into new material. As a last step, creators or editors can share original or adapted content with others (*redistribute*).

First, the application of version management to OER is reflected in two use cases before discussing challenges related to scope and type of materials and formats. Using the example of GitHub possible adaptation of version management functions to OER are presented. Furthermore, educational tools applying such functions are described.

3.1 Use Cases of OER and Version Management

When version management is applied to the creation and modification of OER, two use cases can be distinguished. The difference lies in whether the adaptations are made to a resource or whether further materials are edited by external users independently of the original resource.

New versions of a material are created when content is changed, e.g., by additions or updates. This can be done by one author as well as by several people collaboratively. A clear presentation of the versions with options for comparing versions and changes plays an important role for comprehensibility. Changes to learning resources can be of various types and scopes. Formal changes may be minor corrections of grammatical or spelling errors or linguistic improvements.

Content changes can refer to updates and additions, but also to adaptations to individual contexts with different subject and local requirements. Didactic or technical changes in the arrangement of learning content or the use of tools are also conceivable. These changes can lead to a variety of new versions.

According to the idea of OER, the possibility of using and editing materials of others is a second use case which can be connected with version management. Users can adapt the material of an author to their individual contexts and make it available again so that besides the original resource, derivatives by different users might exist. Version management functions can support this use case by linking the original resource to the modified derivatives. This connection leads to the original resource being linked to the derivative and, at the same time, all derivatives being listed consolidated with the original resource.

3.2 Material Types and File Formats

To apply these use cases to version control, types and formats of materials need to be considered. OER comprise different scopes and types of materials which extend to different levels (Kerres, 2016). Firstly, single materials such as presentation slides, images, audios, videos, exercises, or interactive elements are learning objects that can be directly reused and integrated into materials and used in teaching. Secondly, there are teaching units that consist of a collection of materials, for example, text documents together with exercises, as in an h5p¹ or SCORM element. Also, online textbooks or notebooks, such as Jupyter Notebook,² contain several collected materials. Finally, the third level contains entire courses from a learning management system or a MOOC platform.

These levels of granularity and modularity influence the practical application of OER. Reusing and sharing resources becomes more effective and flexible when learning objects consist of single resources. (Salas, 2020). This is also evident from the fact that teachers prefer to reuse materials with a smaller scope (Schroeder & Krah, 2021). Therefore, it is necessary that OER repositories enable the provision and subdivision of resources into thematic units showing single elements.

Besides material types, open file formats play an important role in OER in the context of version management. On the one hand, the idea of OER is that content can be further processed without technical restrictions using openly accessible

¹ <https://h5p.org/>.

² <https://jupyter.org/>.

tools. On the other hand, open text files enable utilising the advantages of version control, such as the display of differences between versions.

3.3 Principles of Data Versioning

Besides software code, versioning has been applied to several use cases of data management (Klump et al., 2020a). The Research Data Alliance developed principles of data versioning (Klump et al., 2020b) oriented to version control of software code. These principles of data versioning can also be considered relevant for learning resources since they can be revised, released as new versions containing several materials in one resource or different file formats, as well as derived from other resources.

A changed instance of a dataset that is produced during data production is called a *revision*, whereas a *release* indicates a new data product after several revisions during the production of a dataset. The nature and significance of the change should be described. As part of the data versioning principles data repositories should consider different *granularities* and *manifestations* of data. Datasets may be combined into collections containing different sub-collections. Therefore, both granularities, collections, and datasets, need to be identified and versioned. Likewise, the same dataset may be occurred in different file formats so that the same content has different manifestations that need to be identified and connected. Furthermore, a release should contain information on its provenance when it is derived from other data products.

3.4 Version Management Functions for OER

GitHub, as a platform using functions of Git, enables version management during collaborative software development. GitHub functions can be presented as a workflow (see Fig. 1), where a creator initiates a project for producing material (black process). This content can be modified by contributors as part of the project and fed back into the main material (blue process). In addition, a material can be copied by other users, modified, and reported back to the original author (green process).

In addition to making source code available, GitHub can also be used for version management of documents, so that application scenarios for OER are possible. Some processes presented below can be transferred to different scenarios for educational resources (see Table 1).



Fig. 1 GitHub workflow of selected processes. (Own illustration)

The main function of version control is saving changes with *commits* and maintaining a history of those changes as well as assigning them to authors. This is essential, especially for the development of software, to be able to revert to

Table 1 Git(Hub) functions and transfer to OER (Own illustration)

Function	Relevance for software development	Transfer to educational resources
Release	Providing an interim state/stable version	Providing updated or corrected content as a new version
Commit	Save changes	Overview and transparency of changes
Diff	Display and comparison of differences and deviations (esp. for source code)	Display and comparison of differences and deviations (esp. for text files)
Fork	Splitting off a project for own development	Copy a resource for reuse
Branch	Ramification within a project for separate development	Subdivision of a resource into individual elements
Pull Request	Returning improvements or further developments to maintainer	Returning improvements or further developments to creator of a resource
Merge	Accepting changes and joining requests to original project	Accepting changes and joining requests to original resource

a previous, stable version in case of identified errors. However, individual or several authors working collaboratively with documents also benefit from considering older versions of a resource and restoring them if necessary. Change comments, known as *commit messages*, can be a valuable support in tracking changes between versions. However, social coding generates many more changes than revisions or adaptations of learning resources. Especially providing several revisions consolidated in a new version in as a *release* seems to be relevant for OER (see Sect. 3.3) and can be associated to the first use case (see Sect. 3.1). To illustrate the types of changes that have occurred among versions for users, the concept of semantic versioning (Preston-Werner, n. d.) uses a notation of three digits, e.g., 1.2.3. The first digit indicates major revisions, e.g., incompatible API changes. Minor revisions are made by adding functionalities to a new version, marked by increasing the second digit. Corrections such as bug fixes are tagged by a patch with the third digit.

With the *Diff* functionality, deviations in text files are displayed character by character, which is indispensable for the traceability of source code. Especially in the collaborative creation of learning resources, viewing exact deviations can

contribute to quality control. However, OER are predominantly binary files, e.g., images, which are formatted and need to be interpreted by a programme. In contrast, text files just need a simple editor to be readable. That is why conversions from binary files to text files would be necessary, for example via Pandoc,³ to be able to compare content of different files. This option exists for documents (docx, odt), books (epub) or tables (csv) with the target format Markdown, among others. However, this option is not feasible for many file formats. Special programmes such as pdftotext offer the conversion option for other file formats, but these are lossy and only give a rough overview of changes (Haenel & Plenz, 2014). For other file formats, such as image, video, and audio files, matching is technically possible but resource-intensive, for example, transcribing spoken audio content and comparing the transcripts automatically to highlight the differences.

A *branch* creates ramification within a project, creating different working environments for developments that can be fed back into the project (see Fig. 1, blue process). This functionality can be applied to learning resources when a material can be divided into individual elements in terms of a smaller granularity, for example, book chapters or learning units of a course. These elements can be edited or added in a single branch by individual users and integrated into the entire material if necessary.

A project can be split off to expand or test one's own development based on the code. The copy remains linked to the original so that further developments can be displayed with the original record. This *fork* method can be adapted for learning resources to create the basis for a derivative as described in the second use case (see Sect. 3.1). In this way, a material can be adapted to individual contexts (see Fig. 1, green process). Authors can trace who copied and edited their materials and other users have an overview of further versions of this material.

Pull requests can be used so that editors can inform the creator of the original resource about changes. It is up to the creator of the project to decide whether they want to integrate the improvements or adaptations into their materials and versions in the sense of a *merge*. In this way, learning resources can be linked or integrated into each other, for example, in the case of independently created text parts.

³<https://pandoc.org/>.

3.5 Educational Tools Using Version Management Functions

For a first insight into possibilities of transferring and using version management functions for OER, tools used in the educational context were analysed. It was found that these tools used for different material types and scenarios partly applied version management functions (see Table 2).

Collaborative writing and editing tools like Etherpads, e.g., HackMD, or Wikis provide an overview of version history and change messages as well as comparisons of differences. For example, Wikibooks enables the collaborative creating of open textbooks based on the Wiki software Mediawiki.

Several learning resources adapt the fork method to allow users to copy resources of others and modify or develop content for their individual needs and ideas. Examples for tools applying these functions are Scratch for learning pro-

Table 2 Tools for creating and editing learning materials

Tool	Link	Description	Collaborative editing	Version history + commit messages	Diff	Fork
Wikibooks	https://de.wikibooks.org/wiki/Hauptseite	Open Textbooks	x	x	x	
HackMD	https://hackmd.io/	Etherpad	x	x	x	
Scratch	https://scratch.mit.edu/	Programming				x
LearningApps	https://learningapps.org/	Interactive learning content				x
GeoGebra	https://www.geogebra.org/	Apps for Maths				x
SlideWiki	https://slide-wiki.org	Presentation files	x	x		x
Tutory	www.tutory.de	Work sheets				x
Memucho	www.memucho.de	Exercises	x	x	x	x

gramming, Learning Apps for interactive learning content, SlideWiki for presentation files, Tutor for work sheets, or Memucho for learning exercises.

Some OER-repositories in Germany are based on the software edu-sharing⁴ which, like others, focusses on storing resources rather than collaborative editing or exchanging content. New versions can be created for authors in their workspace, viewed, and restored if necessary. The last modification date gives external users an indication of new versions, but further modification details are not visible.

4 Version Management in a Distributed OER Infrastructure

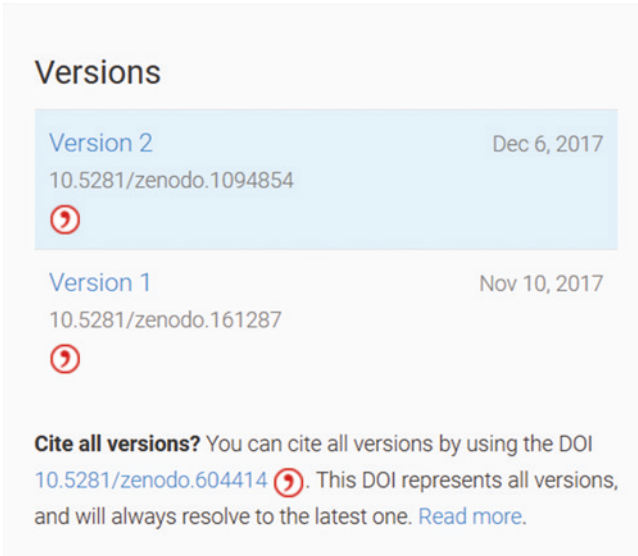
Transferring version management functions to OER can be included in a concept of version management for OER in a distributed infrastructure which addresses issues such as metadata and persistent identifiers, tracking changes, further developments, as well as availability of new versions. Selected functions for managing versions and reusing, editing, and sharing materials are presented below. Special attention is paid to specifications of a distributed infrastructure that focus on discoverability and display of materials and their versions. Besides, it needs to be considered that a distributed infrastructure is based on the collection of metadata from various sources and repositories containing materials. Therefore, no content or files are available on a central platform, so that some processes of version management cannot be implemented as on an individual platform.

4.1 Persistent Identifiers


Persistent Identifiers (PIDs) are established for scientific publications, but OER are not usually assigned to this category. Also, the concept of OER, changeability through edits and adaptations does not correspond to long-term availability and permanent accessibility at first glance. However, to enable referencing and linking between repositories in a distributed infrastructure, PIDs are a necessary and useful integration.


Digital Object Identifiers (DOI) are widely used for scientific articles to permanently refer to digital objects. A DOI assigned to a digital object will remain

⁴<https://edu-sharing.com/produkt/>.



Versions

Version 2 Dec 6, 2017
10.5281/zenodo.1094854


Version 1 Nov 10, 2017
10.5281/zenodo.161287



Cite all versions? You can cite all versions by using the DOI [10.5281/zenodo.604414](https://doi.org/10.5281/zenodo.604414) . This DOI represents all versions, and will always resolve to the latest one. [Read more.](#)

Fig. 2 Example for Zenodo DOI-Versioning. (Zenodo website of Czerniewicz et al., 2017)

throughout the object’s existence, even when the location changes. As DOIs are meant to be permanent, they cannot be changed or deleted. Using the state “registered”, the DOI will not be found unless someone knows the exact DOI string (DataCite, 2020).

In order to reference to individual versions as well as to connect all versions to one resource, the Zenodo platform uses a versioning concept (see Fig. 2) in which each version is assigned a DOI and the entire work is assigned a “concept DOI” that refers to all versions (Zenodo, n. d.).

Zenodo’s DOI versioning appears to be a good way of representing different versions of OER, in that both individual versions and the entire dataset as a total resource receive a DOI. Adaptations of teaching materials lead to a large number of versions that are classified as no longer up to date or incorrect and would normally be deleted so that they do not remain in circulation. However, PIDs ensure that they and associated content cannot be deleted. One possible solution is to archive these versions so that the DOI remains up to date but is redirected to the “concept DOI” and other available versions can be accessed here. This solution can also be an option for users who want to assign a DOI only to selected versions. Thus, it could be integrated as an optional feature.

Following this concept, various versions would be consolidated in a dataset of one resource so that users have a better search experience as they do not find several similar results for one resource. It makes sense to find only the most recent version and display it in the results list. As a requirement for search functions, it is therefore necessary that older versions are no longer integrated in the search index.

4.2 Metadata

Metadata for OER are based on standards such as Dublin Core, a general description of electronic resources, as well as LOM (Learning Object Metadata) for the description of learning objects. With regards to versioning, Dublin Core offers the property *relation* (DCMDI, 2021) with sub-properties such as *has Version / is Version Of* and *references / is Referenced By*. LOM contains *version* within the element *lifecycle*, which describes properties of the history and current status of the learning object and identifies the people and organisations involved in its creation (IEEE Std, 2020). Whereas *version* in LOM presents the status in terms of a version number, Dublin Core allows specifically linking of versions and records. Therefore, Dublin Core provides appropriate elements for applying version management.

New versions of one's own material can be linked to the metadata field *has Version – is Version Of*. Since some OER repositories, such as the edu-sharing software, offer the function of uploading a new version, these can also be included in a distributed infrastructure. Other repository software does not offer this function of adding a new version to a record. Instead, users can create a new record and link it to the previous source record with appropriate metadata. This new record needs to be recognised as a new version of the original record and mapped to it which can be seen as a requirement for a distributed infrastructure.

To connect the derivative with the original resource, the metadata field *references / is Referenced By* can be used as a link. As this possibility is missing in connected systems, the distributed infrastructure has to serve as a central place to hold this information. Describing and connecting underlying resources of a remix with metadata appears to be a complex matter that cannot be adequately mapped in a distributed infrastructure, so this has not been considered further in this context.

In case a new version or derivative is uploaded a second time in another external system, the infrastructure would need to have a duplicate check, which is not exclusively an issue of version management, but rather a general concern of mapping records of different sources in a distributed infrastructure environment.

4.3 Concepts and Functions of Version Management

As previously shown, functions and processes of version management can partly be adapted for OER (see Sect. 3) and are, therefore, integrated into this approach. Moreover, this concept is based on empirical findings from an interview study with higher education teachers aimed at identifying practices and behaviours in working with OER where requirements and relevant functions for OER infrastructures can be derived from (Schroeder & Krah, 2021).

In contrast to version control in software development, changes to educational material are not usually made live in online-editors. Rather updated material is uploaded as a new file. Therefore, releases can be seen as new versions provided with a new PID rather than single modifications as revisions (see Sect. 3.3). Releases can serve as basis for a version history to obtain an overview of different versions of a resource. This includes information such as DOI, person, date, and details of version changes (see Fig. 3). As described, the concept comprises that the entire resource receives a DOI automatically, while authors can optionally assign a DOI when uploading a new version. Version numbers can be realised as single counted digits rather than using the concept of semantic versioning. Since not every single change is saved as a new release, the transparency of changes is saved and realised through commits.

To be able to track differences between versions, change comments serve as a reference point (Schroeder & Krah, 2021). Authors can add these *commits* as

Description		Version History		Further Versions	Sources	Discussion
New Version		Archive Version		Open		Download
Select	Versions	Date	Persons	Files (all) ▼	Modifications	
<input type="checkbox"/>	Version 3 https://doi.org/10.abcdef_3	30.01.2021	Olli Openness	CreatingOER_3.pptx CreatingOER_3.md	Formal corrections	
<input type="checkbox"/>	Version 2 https://doi.org/10.abcdef_2	10.09.2020	Olli Openness	CreatingOER_2.pptx CreatingOER_2.md	Content update	
<input type="checkbox"/>	Version 1	05.08.2020	Olli Openness	CreatingOER_1.md	First Draft	
Resource DOI: https://doi.org/10.abcdef						

Fig. 3 Version history. (Own illustration)

Description		Version History	Further Versions	Sources	Discussion
New Version		Archive Version			
Select	Versions	Person	Date	Files	Modifications
<input type="checkbox"/>	Version 1 https://doi.org/10.abcdef_1	Toni Testing	18.04.2021	20210318.pptx	Added own version
Resource DOI: https://doi.org/10.abcdef					
Original Resource: Creating OER/ Olli Openness (Version 3)					

Fig. 4 Added derivative. (Own illustration)

release messages while uploading a new version. Tracking character-specific changes using the function *Diff* to compare two versions with colour markings are hardly to realisable in a distributed infrastructure as content is not stored.

According to the Data Versioning Principles, granularity and manifestation of materials and files need to be considered. Therefore, a filtering option within each dataset might be a possibility for individual repositories. Especially for materials with a larger scope, it is possible to select different material types to show a subdivision into separate content elements. In addition, uploading different file formats is relevant.

Higher education teachers reported that they were very interested in learning about the external use, editing and further dissemination of their materials (Schroeder & Krah, 2021). To be able to trace which derivative is based on which resource, it is important to maintain a connection to the original material. The GitHub-function *Fork* can be used for this process (see Sect. 3) and is realised in this concept. Applying this, an editor copies a resource into a new dataset by clicking on “Add own version”, where the connection to the original resource remains visible and own versions can be added and uploaded by the new owner of the resource (see Fig. 4). At the same time, this new resource is added to the original under “Further versions” in order to provide an overview of reused and edited materials. This can give both the original author and external visitors indications about further developments and possible uses (see Fig. 5).

Version control also offers functions for cooperative development and usage of content, for example, returning improvements to the creator of a resource via pull requests or merging requests to the original resource. These aspects are more

Description		Version History	Further Versions	Sources	Discussion
Add own version					
Person	Title	Date	Comment	Basis resource	
Testing, Toni	Creating and revising OER	18.04.2021	Additions to content	Creating OER / Olli Openness (Version 3)	
New, Nina	Create OER for biologists	12.12.2020	Extension for target group	Creating OER / Olli Openness (Version 2)	

Fig. 5 Overview of derivatives. (Own illustration)

relevant for a stand-alone platform, therefore, they are not focused on in presenting a distributed infrastructure.

4.4 Availability of New Versions and Derivatives

When authors upload new versions, the difficulty is how users become aware of this since they usually download and store resources in personal environments rather than consulting repositories or websites. Especially when thinking of a distributed infrastructure, finding out about new versions and derivatives is a major problem because resources are stored in disseminated repositories and users may no longer be aware of their place of origin or do not look for the specific dataset. However, it may be interesting for users to know if new versions exist of a material. A sensible solution for this concern will be a central contribution to realising version management for OER in a distributed infrastructure.

In the context of scientific publications, Crossref has addressed these issues with its service Crossmark, where a “button gives readers quick and easy access to the current status of an item of content, including any corrections, retractions, or updates to that record” (Vickery, 2020). This allows users to identify published versions of scholarly content. Readers click on the button *Check for updates* on the publisher’s website or within the PDF file, whereupon a popup box appears showing the current status of the document. In case an article has updates, a Crossref DOI link directs readers to the current version on the publisher’s website (Meyer, 2011). When a correction replaces the earlier version completely, the DOI of the corrected content will be the same as the DOI for the original Crossref deposit (Lamney, 2014).

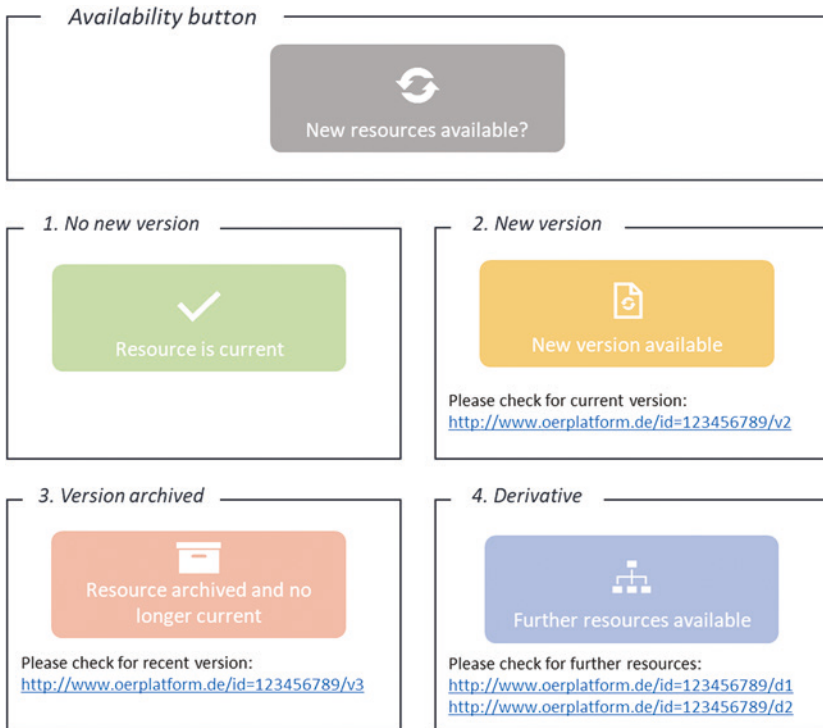


Fig. 6 Availability of new versions and derivatives. (Own illustration)

This concept can be seen as a solution to be adopted for OER, when users check if new versions or derivatives of a resource are available and are directed to the appropriate repository the resource is stored in. An adaptation of the Cross-mark button can be integrated into suitable materials as well as into datasets in case versions of a resource are stored in different locations. By clicking on the “availability button”, four different views related to four use cases may appear (see Fig. 6). A resource can be original and current so that no further updates will be indicated (1.). If a new version is available, a link is added to get to the corresponding resource in the current version (2.). For an archived version, which is no longer available, a link to the most recent version is provided (3.). This check for the availability of new versions can also be applied for derivatives. In this case, all connected resources are linked (4.). This concept enables users to check directly

against the material at hand whether new versions or derivatives are available, regardless of their location. Therefore, this contribution provides a solution for managing versions and derivatives in a distributed infrastructure.

5 Conclusion

This contribution discussed use cases of OER in the context of version management and presented approaches to managing educational materials in a distributed infrastructure, resulting in a concept of version management for OER. It could be shown that version management functions and processes from software development are already partly adapted by some educational tools and can be transferred to OER. In addition, concepts related to the publication of scientific articles and the management of research data may also be applied to OER. The presented ideas on version management for OER represent an approach for further development, taking into account the challenges of a distributed infrastructure.

Overall, infrastructures need to be well designed to make it easy for users to find and share OER in higher education. Especially, the exchange of materials and cooperation in communities should be given greater focus and support by infrastructures.

References

- Atenas, J., & Havemann, L. (2014). Questions of quality in repositories of open educational resources: A literature review. *Research in Learning Technology*, 22, 1–13. <https://doi.org/10.3402/rlt.v22.20889>
- Brooks, C., Cooke, J., & Vassileva, J. (2003). Versioning of learning objects. In *Proceedings 3rd IEEE international conference on advanced technologies* (pp. 296–297). <https://doi.org/10.1109/ICALT.2003.1215091>
- Clements, K., Pawlowski, J., & Manouselis, N. (2015). Open educational resources repositories literature review – Towards a comprehensive quality approaches framework. *Computers in Human Behavior*, 51, 1098–1106. <https://doi.org/10.1016/j.chb.2015.03.026>
- Czerniewicz, L., Deacon, A., Walji, S., & Glover, M. (2017). OER in and as MOOCs. In C. Hodgkinson-Williams & P. B. Arinto (Eds.), *Adoption and impact of OER in the Global South* (pp. 349–386). <https://doi.org/10.5281/zenodo.1094854>.
- DataCite. (2020). DOI states. <https://support.datacite.org/docs/doi-states>, Accessed: 22. Nov. 2021.
- DCMDI. (2021). Dublin Core Metadata Initiative: Relation. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/elements11/relation/>. Accessed: 22. Nov. 2021.

- Franzetti, C. (2019): *Essenz der Informatik*. Springer Vieweg
- Haenel, V., & Plenz, J. (2014). *Git – verteilte Versionsverwaltung für Code und Dokumente*. Open Source Press.
- Heck, T., Kullmann, S., Hiebl, J., Schroeder, N., Otto, D., & Sander, P. (2020). Designing open informational ecosystems on the concept of open educational resources. *Open Education Studies*, 2(1), 252–264. <https://doi.org/10.1515/edu-2020-0130>
- IEEE Std. (2020): 1484.12.1-2020 - IEEE Standard for Learning Object Metadata. <https://doi.org/10.1109/IEEESTD.2020.9262118>.
- Keres, M. (2016). Open educational resources (OER). In N. Gronau, J. Becker, E. J. Sinz, L. Suhl, & M. Leimeister (Eds.). *Enzyklopädie der Wirtschaftsinformatik* (9. Aufl.). <http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon>. Accessed: 22. Nov. 2021.
- Keres, M., Hoeltherhof, T., Scharnberg, G., & Schroeder, N. (2019). EduArc. Eine Infrastruktur zur hochschulübergreifenden Nachnutzung digitaler Lernmaterialien. *Synergie. Fachmagazin für Digitalisierung in der Lehre*, 7, 66–69. <https://doi.org/10.25592/issn2509-3096.007.014>
- Klump, J., Wyborn, L., Downs, R., Asmi, A., Wu, M., Ryder, G., & Martin, J. (2020a). Compilation of data versioning use cases from the RDA data versioning working group. Version 1.0. Research Data Alliance. <https://doi.org/10.15497/RDA00041>.
- Klump, J., Wyborn, L., Downs, R., Asmi, A., Wu, M., Ryder, G., & Martin, J. (2020b). Principles and best practices in data versioning for all data sets big and small. Version 1.1. Research Data Alliance. <https://doi.org/10.15497/RDA00042>
- Lamney, R. (2014). How to apply CrossMark and FundRef via CrossRef extensible markup language. *Sci Ed*. 2014;1 (2): 84–90 <https://doi.org/10.6087/kcse.2014.1.84>
- Meyer, C. A. (2011). Distinguishing published scholarly content with CrossMark. *Learned Publishing*, 24, 87–93. <https://doi.org/10.1087/20110202>
- Ovadia, S. (2019). Addressing the technical challenges of Open Educational Resources. *Portal: Libraries and the Academy*, 19(1), 79–93. <https://doi.org/10.1353/pla.2019.0005>.
- Preston-Werner, T. (n. d.). Semantic Versioning 2.0.0 <http://semver.org>. Accessed: 22. Nov. 2021.
- Salas, R. P. (2020). Reusable learning objects: An agile approach. *IEEE Frontiers in Education Conference (FIE)*, 2020, 1–6. <https://doi.org/10.1109/FIE44824.2020.9273947>
- Santos-Hermosa, G., Ferran-Ferrer, N., & Abadal, E. (2017). Repositories of open educational resources: An assessment of reuse and educational aspects. *The International Review of Research in Open and Distributed Learning*, 18(5), 84–120. <https://doi.org/10.19173/irrodl.v18i5.3063>.
- Schroeder, N., & Krahl, S. (2021). Anwendung von Open Educational Resources in der Hochschullehre. In H.-W. Wöllersheim, M. Karapanos, & N. Pengel (Eds.), *Bildung in der digitalen Transformation* (pp. 121–130). Waxmann. <https://doi.org/10.31244/9783830994565>.
- Schroeder, N., & Pfaender, P. (2020). Nutzung von GitHub für Open Educational Resources. In R. Zender, D. Ifenthaler, T. Leonhardt, & C. Schumacher (Eds.), *DELFI 2020 – Die 18. Fachtagung Bildungstechnologien der Gesellschaft für Informatik e.V.* (pp. 337–342). Gesellschaft für Informatik e.V., Bonn. <https://dl.gi.de/handle/20.500.12116/34180>.
- Vickery, B. (2020). Crossmark. <https://www.crossref.org/services/crossmark/>. Accessed: 22. Nov. 2021.

- Vijayakumaran, S. (2019). *Versionsverwaltung mit Git*. mitp Verlag.
- Wiley, D. (2014). The access compromise and the 5th R. Iterating toward openness. <http://opencontent.org/blog/archives/3221>. Accessed: 22. Nov. 2021.
- Zagalsky, A., Feliciano, J., Storey, M.-A., Zhao, Y., & Wang, W. (2015). The Emergence of GitHub as a collaborative platform for education. In D. Cosley, A. Forte, L. Ciolfi, & D. McDonald (Eds.), *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing* (pp. 1906–1917). <https://doi.org/10.1145/2675133.2675284>.
- Zenodo. (n. d.). DOI Versioning. <https://help.zenodo.org/#versioning>. Accessed: 22. Nov. 2021.
- Zervas, P., Alifragkis, C., & Sampson, D. G. (2014). A quantitative analysis of learning object repositories as knowledge management systems. *Knowledge Management & E-Learning*, 6(2), 156–170. <https://doi.org/10.34105/j.kmel.2014.06.011>.
- Zolkifli, N. N., Ngah, A., & Deraman, A. (2018). Version control system: A review. *Procedia Computer Science*, 135, 408–415. <https://doi.org/10.1016/j.procs.2018.08.191>

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

