Electronic Communications of the EASST
Volume 85 Year 2025

deRSE25 - Selected Contributions of the 5th Conference for Research Software Engineering in Germany

Edited by: René Caspart, Florian Goth, Oliver Karras, Jan Linxweiler, Florian Thiery, Joachim Wuttke

Reuse and bidirectional further development of research software in a Digital Humanities project

Dennis Ried

DOI: 10.14279/eceasst.v85.2682

Reuse and bidirectional further development of research software in a Digital Humanities project

Dennis Ried[1]

[1] dennis.ried@musikwiss.uni-halle.de, http://www.dennisried.de/
Institut für Musik, Medien- und Sprechwissenschaften
Martin-Luther-University Halle-Wittenberg

Abstract: This article presents a workflow concept developed at the Henze-Digital (HenDi) project (2021–2024, Paderborn University), aiming to create a digital letters edition. Methods and technologies from the Carl Maria von Weber-Gesamtausgabe (WeGA) have been reused, particularly its research software and TEI data schemas. The article focuses on the method of software reuse and TEI schema adaptation using ODD-Chaining, which enabled leveraging WeGA's structures while tailoring them to HenDi's needs. It also discusses challenges in software reuse, such as maintaining a stable connection between a Research Software and its derivate, and presents the project's software-chaining approach–a transfer of the ODD-Chaining-concept–as one possible solution. The article offers insights into the benefits and challenges of reusing methods and technologies in digital humanities projects, emphasizing the importance of collaboration and flexible software development approaches.

Keywords: Digital Humanities, Software Development, Research Software, Code Sustainability, Workflows

1 Introduction

From 2021 to 2024, the infrastructure for a digital edition of the letters of Hans Werner Henze[1] was created at the University of Paderborn (Germany). As part of this project funded by the German Research Foundation (DFG[2]), approximately 600 documents were to be edited within three years using the TEI-XML standard and made available in Open Access [BBC+24, 4] [CMR24a].

Since Henze-Digital (HenDi)[3] was not the first digital letter edition and the capacities in a short-term project were limited[4], a focus on reusing existing methods and technologies was already placed during the project application. This approach was considered novel in the field of musicology.

The technological foundation was based on the research software (WeGA-WebApp [SSZ+25]) and data schemas (WeGA-ODD [SNS+25]) of the Carl Maria von Weber-Gesamtausgabe

[1] 1926–2012, a renowned German composer in music of the 20th century, who gained international recognition primarily through his opera compositions and symphonies.

[2] Official title: "Digital letter edition: Hans Werner Henze's network of artists, short title: Henze digital", https://gepris.dfg.de/gepris/projekt/459602398.

[3] Hans Werner Henzes künstlerisches Netzwerk, https://henze-digital.zenmem.de/.

[4] One position each for the Edition (full-time) and the Digital Humanities (part-time, 65%).

(WeGA)[5], which has been developed for over ten years (since 2011) in Detmold/Paderborn (i.e., at the same location) and is recognized as "state of the art" in musicology. Thorsten Röder described the Weber letter edition–a digital aspect of the WeGA project–in 2020 as defining the "state of the art" in digital letter editions [Rö20]. The decision to reuse existing technologies was based on WeGA's long-standing experience with TEI-XML and the high recognition of their research software. Due to the connection of both projects to the same institution, there was full access to the respective infrastructure, which facilitated the reuse of technologies and processes, as well as the development of a practical workflow in many respects.

Another software that was available for selection was, e.g., the TEI Publisher[6]. An open source software whose customizations (of the appearance of the text) are done via the data schema (TEI-ODD, Subsection 2.1). The TEI Publisher is a powerful, detailed software solution for text editions. The focus on text editions was ultimately the strongest argument for not using this software in the context of HenDi. After all, it was already clear that some data records would have to be created in MEI[7]. On the other hand, it should be possible, e.g., to potentially compile the basis of an edition of Henze's collected musical works, or a catalog of works using the data of this letter editions at a later date. With the WeGA-WebApp–where the usage of MEI is implemented broadly–all of the set goals, as well as possible future goals, seemed achievable.[8]

The thesis was that the WeGA-WebApp, which could already process the research data of the WeGA, would only require project-specific adaptations for Henze-Digital. This promised a swift start to the editorial work (transcription of documents and development of a critical commentary). Project-specific adaptations were necessary, primarily because there is a historical distance of more than 200 years between the birth of Carl Maria von Weber (1786–1826) and the death of Hans Werner Henze (1926–2012), during which the media used for letters have changed significantly (e.g., the invention of the telegram, fax, and e-mail).

The WeGA-WebApp is a web application for eXist-db[9]–a native XML database. At its core, the software consists of XQuery modules and XSLT scripts for the data processing. Since the results of the XQuery functions can be stored in maps (JSON), the WeGA-WebApp uses HTML templates for presenting the processed data. Using eXist-db's HTML-templating module[10] provides the definition of layout templates that can be filled in with the data from JSON maps. With this technique, the presentation layout becomes uniformly because the data processing output is structured data and does not necessarily consist of HTML fragments.[11]

Since HenDi and WeGA were projects that were conducted simultaneously, a way had to be found to reuse and further develop a research software that is constantly being developed.[12] From

[5] Carl Maria von Weber-Gesamtausgabe, https://weber-gesamtausgabe.de/.
[6] TEI Publisher. The Instant Publishing Toolbox, https://teipublisher.com/.
[7] Musical content and its metadata is usually recorded using the Music Encoding Initiative, https://music-encoding.org/.
[8] The TEI Publisher and also the Edirarum-Framework for the OxygenXML Developer (https://www.ediarum.org/index.html) did not fit the use of MEI in the project's context.
[9] eXist-db. The Open Source Native XML Database, https://exist-db.org/.
[10] HTML Templating Module, https://exist-db.org/exist/apps/doc/templating.
[11] Concerning software dependencies, Subsection 3.3.
[12] The inspiration goes back to Peter Stadler (creator of the WeGA-WebApp, ORCID: 0000-0002-2544-1481), who came up with the idea early that new features and improvements emerging in the context of Henze-Digital could ideally also flow back into the origin (WeGA-WebApp).

this basic constellation, the premise arose that the development of a HenDi-WebApp [Rie24] must be designed in such a flexible way that the return of features to the original source code always remains possible. Ultimately, both projects would benefit from such an approach.

In this article, aspects of the reuse and further development of research software that have emerged in Henze-Digital are reported and reflected upon. To explain the methodological approach of software reuse, a look at the adaptation of TEI data schemes must first be taken.

2 Adaptation of TEI-ODD-Schemas

The encoding and editing guidelines of the WeGA are reflected in the project's TEI-ODDs [SNS$^+$25], which are project-specific customizations of the TEI standard [Con25a].

2.1 TEI-ODD

The abbreviation ODD stands for "One Document Does it all" and refers to a technology provided by the Text Encoding Initiative (TEI).[13] ODD itself is a TEI-XML format used to define a schema for TEI. Essentially, an ODD document is a blueprint for a compilable XML schema.[14] With TEI-ODD, it is possible to define entities (elements, attributes, modules, classes, and macros) and document their usage in a single file. The human-readable documentation can take the form of explanations within the definitions (describing the meaning and application of the defined elements and attributes, etc.) or as prose text that allows the integration of the project's editing and encoding guidelines [Con25a].

The TEI itself is defined in ODD, and the TEI schema is designed to be used as a set of building blocks for creating a specific project schema [Con25a]. Due to the highly differentiated TEI standard, it makes sense to restrict, adjust, or extend the standard to the conditions of one's own project.

2.2 ODD-Chaining

ODD-Chaining is a methodology that enables the use of an existing ODD specification as a starting point for further adaptation and refinement (i.e., almost a customization of an existing TEI customization). "An ODD file specifies a particular view of the TEI, by selecting particular elements, attributes, etc. from the whole of the TEI. But you can also refine such a specification further, making your ODD derive from another one. In principle, you can chain together ODDs in this way as much as you like" [Bur16].

While the WeGA builds its ODDs on the full TEI specification, the ODDs of Henze-Digital specify the WeGA's ODDs[15] as their source (Figure 1) and can thus be further specified: adding restrictions, deleting elements, adding previously removed attributes, expanding closed value

[13] See chapters "24.3 Customization" and "24.5 Implementation of an ODD System", in [Con25b].

[14] Compiling the ODD can be done using the TEI-Stylesheets (https://github.com/TEIC/Stylesheets) on the command line (depends on Saxon HE with XSLT support, https://github.com/Saxonica/Saxon-HE), the TEIGarage Conversion tool (https://teigarage.tei-c.org/) or within the OxygenXML Editor software (https://www.oxygenxml.com/, commercial product), where the TEI environment is implemented.

[15] The source has to be a compiled ODD.

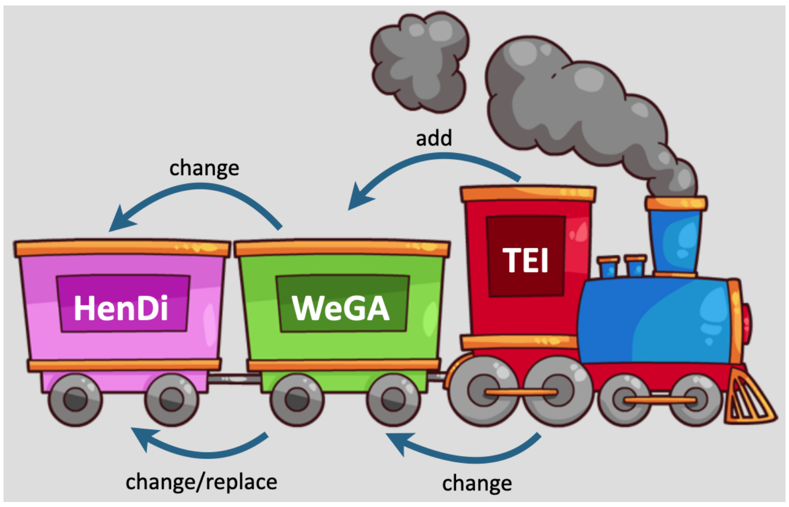

Figure 1: Method of TEI-ODD-Chaining at HenDi [Rie25, 7]

lists, defining new data types, and more. When compiling the (HenDi-)ODD, the set references are resolved, and an output is generated that contains the desired specifications [CMR24b].

As mentioned earlier, it is possible to add elements, modules, and more using this technique. These can be entirely new project-specific elements or existing TEI elements not included in the source ODD. The latter only requires that the reference to these new elements or modules contains its own source statement (`@source`), pointing to the source where these definitions can be found (e.g., the complete TEI P5 specification in a specific version) [Bur16]. This allows combinations of any number of ODDs, as Burnard notes.

However, the applicability of ODD-Chaining also depends on the architecture of the source ODD. To illustrate this, if an element's attributes are added or removed through the element's membership in an attribute class, it is easier to modify them at a later level than if the attributes are explicitly added or removed in the context of a specific element, e.g., as a direct child. Thus, a well-structured ODD source with clear specifications facilitates chaining. This could be supported in the future by the ODD schema, e.g., using Schematron[16] validation to provide hints.

Henze-Digital applied this technology, adapting most of the WeGA's document types and their schemas (for letters, bibliographic records, and reference data sets for persons, organizations, and places). However, necessary modifications and exceptions were made, such as in the treatment of the document type 'documents'. Unlike the WeGA, Henze-Digital not only recorded metadata and transcribed these text documents but also edited them, so the schema HenDi-Documents had to be derived from the WeGA's letter schema (WeGA-Letters) analogous to the schema of the document type letters (HenDi-Letters) (Figure 2). The decision to capture literary works in TEI and record musical and film works with MEI also required adapting the source schemas (Figure 2).[17]

[16] Schematron. A language for making assertions about patterns found in XML documents, https://schematron.com/.
[17] In the WeGA, reference data sets for works were captured with MEI, as the catalog originally only contained musical works. However, it was already known at the beginning of Henze-Digital that literary works would become relevant in significant numbers and had to be captured appropriately with TEI.

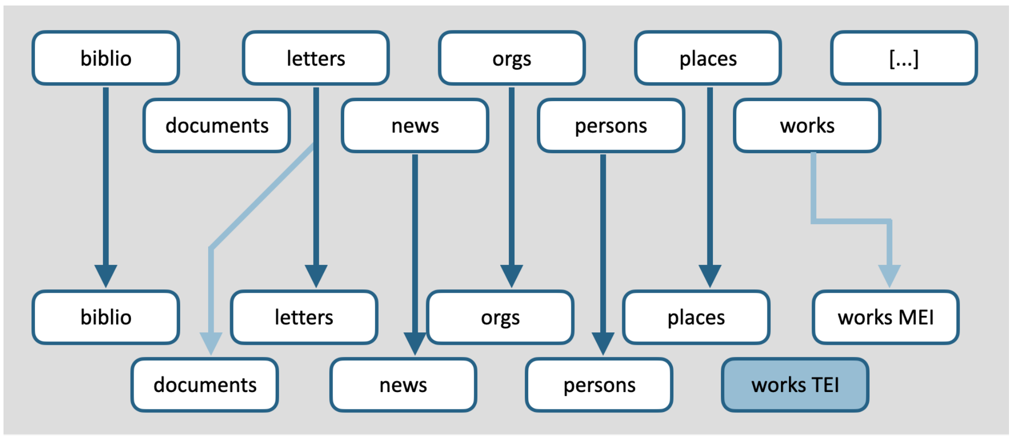

Figure 2: Reusing TEI-ODD-Schemas at HenDi [Rie25, 8]

These differences reflect the diverging research subjects, especially regarding the media landscape. Through ODD-Chaining, HenDi was able to adopt a large part of the WeGA's proven structures–schemas and documentation–while making the necessary adjustments for the specific research subject, while maintaining the basis for data processing through the WeGA-WebApp research software.

3 Reuse of Research Software

Since the WeGA-WebApp [SSZ+25] was already a functional (open source) software that could process the research data of the Weber edition, only adaptations were necessary for Henze-Digital where it deviated from the WeGA's data schemes–at least in theory.[18]

The central challenge was to establish a stable bidirectional connection between the emerging HenDi-WebApp [Rie24] and its parent software, the WeGA-WebApp, while both products were being developed in parallel. This was necessary due to the staggered project timelines, with Henze-Digital scheduled to conclude before the WeGA project is completed. It was therefore expected that new features would emerge or other adaptations would be made in both projects at different times, which had to be responded to. Without an intensive exchange between the projects, parallel development would not have been possible, and the goal of synergy would have been missed.

3.1 First approach – git fork

An initial approach to further develop the WeGA-WebApp's source code in a git fork proved to be unsuitable. More invasive changes to the software architecture made it economically impossible to synchronize upstream and origin. For instance, while the WeGA's identifiers are 7-digit (prefix and decimal `A\d{2}[0-9]{4}`), these were extended to 8 digits (prefix and hexadecimal

[18] The DFG project "Henze-Digital" initially planned not to develop its own software solution for the digital edition and aimed to test the reuse of existing research software instead.

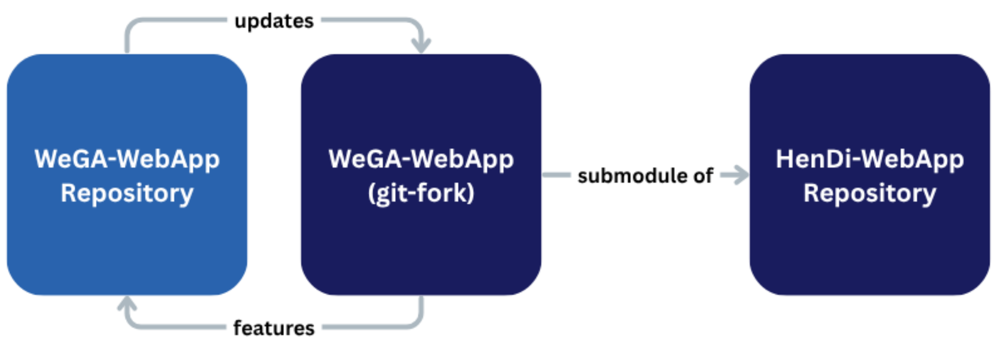

Figure 3: Repository infrastructure

`A\d{2}[0-9A-F]{5}`[19]) in Henze-Digital. Regardless of the advantages and disadvantages that such a fundamental decision may bring, the problem now arose that the software's source code had to be modified in numerous places, which led to enormous conflicts with every attempt to synchronize upstream and origin. These conflicts could be prevented by generalizing the corresponding places in the WeGA-WebApp's source code (upstream), so that the identifier patterns could now be defined as parameters in the configuration file. This modification made an adaptation in Henze-Digital obsolete and at the same time with no impact on the WeGA's operations (Figure 3, left).

However, a fundamental problem remained the fact that the modifications (in XQuery modules, XSLT scripts, JavaScript libraries, SASS files, HTML templates, and the Swagger API) would, due to their increasing number, again cause problems with synchronization. This became particularly apparent when larger updates were made on the WeGA's side in modules that were also modified by Henze-Digital. The result was always conflicts that had to be resolved manually and required an enormous amount of time. A different type of dependency had to be created to maintain a certain level of synchronicity despite the increasing distance from the original source code.

3.2 Second approach – software chaining workflow

In the further course of the project, an attempt was made to transfer the principle of ODD-Chaining to the further development of the research software: defining the WeGA's source code as the base source and specifying the desired changes separately, so that a new software will be distributed after compiling.

The first try to apply this method was to use a compiled version of the source code–analog to the ODD-Chaining concept. This procedure is possible but cumbersome, e.g., when modifications are applied to the SASS files or, in general, when recompiling of some components is necessary or JavaScript libraries need to be added. Thus, the modification must be done before compiling or fetching the JavaScript-libraries to avoid a second step of compiling.

[19] The last position of the identifier is a check-digit. By using XSLT functions in the ODD's constraints, the check digits can be validated by the schema, e.g., by a compiled RelaxNG. See https://github.com/Henze-Digital/HenDi-ODD/blob/e3fc3d1daaef9f3145a0d64c60d8f8612d56e33e/src/Specs/common-specs.odd.xml#L1918-L1967.

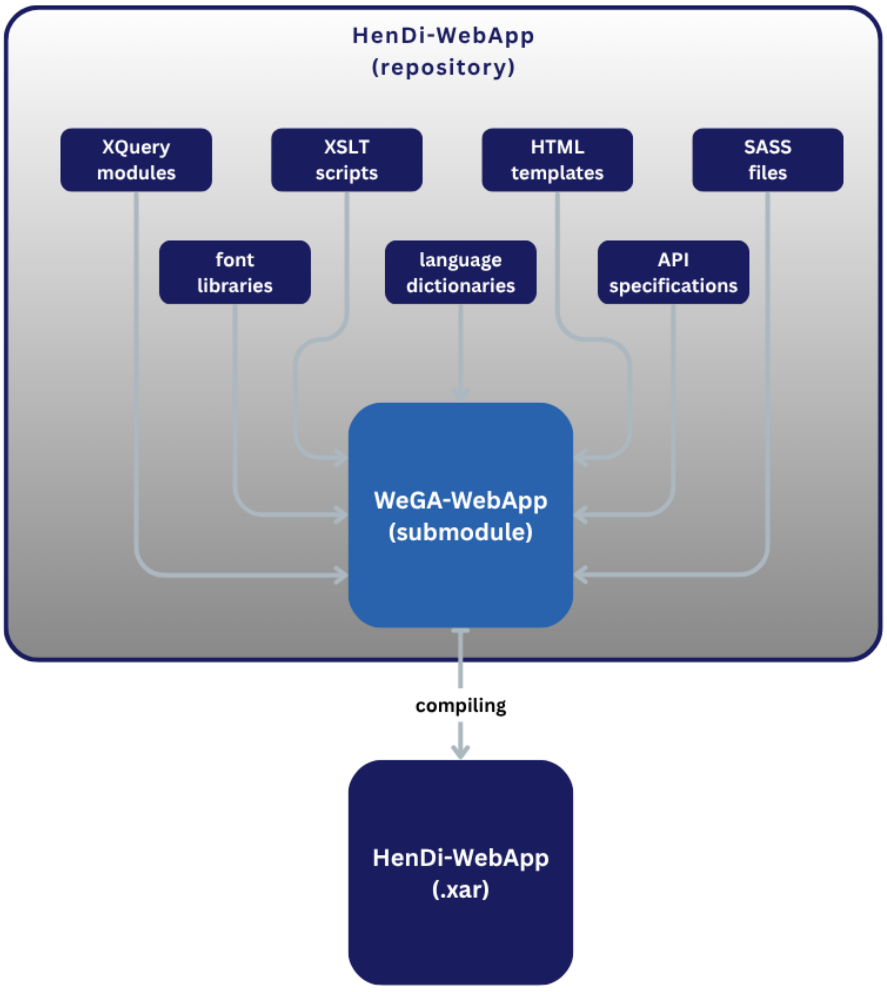

Figure 4: Workflow of overwiriting data and compiling the software

In the final software-chaining approach, a git repository was set up for the HenDi-WebApp, which contains only the project-related modifications to the WeGA-WebApp's source code. The WeGA-WebApp's source code, on which the new software was to be built, was integrated as a git submodule (Figure 3, right).

As a final step, it was only necessary to develop a build script that copies the (new) components from the HenDi repository into the submodule (Figure 4) and compiles the application there.

To simplify the copy tasks, the structure of directories in the HenDi-WebApp repository is analog to the WeGA-WebApp. Thus, all contents in the data directory (Figure 4) could be copied in one step (when overwriting is forced).

As with the WeGA-WebApp, Apache Ant[20] was used as technology for the build script. A second reason to use this technology is that Apache Ant also offers the option of integrating

[20] The Apache Ant project, https://ant.apache.org/.

```
<target name="build-webapp" depends="hendi:init">
    <!-- HenDi-WebApp build preparation -->
    <antcall target="webapp-prepare"/>
    <!-- run sass -->
    <antcall target="wega:sass"/>
    <!-- compile openapi.json-->
    <antcall target="wega:swagger-cli"/>
</target>
```

Figure 5: Reusing includes Ant targets

existing Ant-scripts and to use the included targets with a prefix, allowing existing build routines to be reused (Figure 5).

Since this process was performed in the submodule, the dependencies and paths remained unchanged, so the existing routines could be executed almost without problems.[21]

An advantage of this approach is that only the modules and files that are actually modified need to be maintained. This keeps the number of files to be synchronized manageable. Although this approach greatly relieved the code management, the challenge of keeping the modified code of Henze-Digital and the further developed code of the WeGA-WebApp in sync remained. Even with this method, conflicts can occur when synchronizing, e.g., when the submodule is updated to the latest software version. However, these conflicts arise to a much lesser extent than with the git fork, and the resulting effort was considered acceptable in the HenDi project.

Overwriting entire modules and script libraries is due to the fact that, for example, in XQuery, it is not (natively) possible to override individual functions within the software.[22] Therefore, a module–this also applies to XSLT, etc.–always had to be overwritten as a whole if the output of a single function was to be differentiated differently. (On the other hand, it is precisely this approach that leads to the avoidance of numerous errors or a breakdown when running the software after updating the submodule, since–except for extensive breaking changes–the updated module is overwritten with the former, but modified program code.)

The decisive aspect of bidirectional further development is that new features or also improvements and generalizations of the source code were not only available for Henze-Digital. As long as these could be returned to the WeGA-WebApp or implemented directly, they were immediately available to the WeGA, HenDi, and all other software derivatives. Since these adaptations became automatically available again in the HenDi-WebApp with the next update of the source code (i.e., submodule), the program code in the HenDi-WebApp repository could be reduced by the successfully implemented parts.

Finally, it should be mentioned that if the submodule containing the source code of the soft-

[21] A limitation in reusing an Ant script is its granularity. If changes are necessary in certain processing steps (e.g., because a component needs to be added), these targets must be redefined. The more fine-grained such a script is structured, the easier it is to reuse individual building blocks.

[22] Among other things, it is possible to work with look-up functions, but these must be implemented in the origin source code in order to enable overwriting in a second step.

ware itself is a git fork[23], there is the option of implementing new features there (Figure 3). One effect of this method is a two-stage control process for updates (controlling the git fork and the submodule). On the other hand, this has the decisive advantage that features can be used in one's own project even if the review process for implementation in the original source code has not yet been completed.

3.3 Dependencies

The software dependencies of the WeGA-WebApp are Apache Ant, Saxon, NodeJS, and Yarn. NodeJS and Yarn are used to update JavaScript libraries needed to compile the frontend (e.g., SASS, minify, font packages) and the Swagger API. Apache Ant (and Saxon for additional tasks) is used to define the build workflow and to provide the application as an eXtensible Archive (.xar)[24] for distribution. This archive format can be installed to eXist-db automatically by uploading to the database.[25] However, the software has its dependencies and the chaining-workflow also. In principle, the concept of chaining itself depends on copy tasks–performed here using Apache Ant–and git, since the workflow is designed using git submodules. All other dependencies for building software using this method are determined by the software that is to be reused.

This means that if the software to be chained uses open source technologies, this will also apply to the workflow. If the software conforms to the FAIR4RS principles [CKB+22], the workflow will not change because of that. As long as git is maintained, this workflow should work. The copy tasks should also be able to be performed using technologies other than Apache Ant. From the current perspective, the workflow can therefore be considered sustainable in the long term.

Due to the given dependencies, Apache Ant was used broadly. Using Ant's filtering method allows the creation of templates for the packaging files eXist-db needs (Footnote 24), but also for option files[26], the citation file[27], or other contexts. In the end, the workflow was strongly parameterized with the goal to manage version number updates and setting parameters in the software quite comfortably. Of course, this aspect could be greatly expanded, but this seemed to be of limited benefit for the project at the time.

4 Conclusion

This article has presented the bidirectional development approach implemented in the Henze-Digital project for reusing and extending research software (WeGA-WebApp). Based on ap-

[23] This can also be applied to ODD-Chaining.

[24] The eXist-db's Package Repository (https://exist-db.org/exist/apps/doc/repo) makes it easy to distribute the data to the database. Using the "EXPath Packaging Format" (expath-pkg.xml, repo.xml) provides a xar-archive that can be automatically installed in the database after uploading.

[25] If a Docker image is used for deployment, the data package can be stored in eXist-db's autodeploy directory. On launching the database (i.e., running a new container), the package is installed automatically.

[26] Options in lines 30–35 are written during compilation using the build.properties (https://github.com/Henze-Digital/HenDi-WebApp/blob/e77717b56f69120084dd44340a4170ae94fb47b7/data/catalogues/options.xml.tmpl#L30-L35).

[27] Values in lines 5, 19, 21, 29, 33–34 are written during compilation using the build.properties (https://github.com/Henze-Digital/HenDi-WebApp/blob/e77717b56f69120084dd44340a4170ae94fb47b7/CITATION.cff.tmpl).

proximately eighteen months of productive use, several key lessons can be drawn for future digital humanities projects.

4.1 Technical Lessons Learned

TEI-ODD-Chaining enables sustainable reuse. The concept TEI-ODD-Chaining of proved highly effective. Reusing editing and encoding guidelines developed over more than a decade provided significant benefits. Configuring the source ODD as a git submodule allows for both static and dynamic code management.

We found that TEI-ODD reuse is facilitated when changes are made globally (e.g., by modifying classes) rather than at individual locations (such as removing class membership and adding an attribute directly).

Traditional git forking is unsuitable for architectural changes. Our initial approach of using a git fork proved impractical when fundamental modifications were required. Merge conflicts made synchronization economically unfeasible, demonstrating that simple forking is only suitable for minor customizations.

The workflow is sustainable in the long term. The workflow relies on two widely used, open technologies and can therefore be considered sustainable from today's perspective. The technologies used can be replaced if necessary, especially Apache Ant, which was chosen for project-specific reasons. The more a software project complies with the FAIR4RS principles, the easier it is to reuse it with this workflow.

Software chaining supports maintainable customization. Transferring the TEI-ODD-Chaining concept to software development proved highly effective. By maintaining only modified components separately and integrating the source code as a git submodule, we achieved manageable customization with significantly reduced maintenance overhead.

The git submodule can serve as a stable copy or a dynamic resource controlled by git. If the submodule itself is a git fork, features can be proposed upstream and reused in one's own project simultaneously, reducing dependency on other projects' timelines/capacities.

4.2 Community and Sustainability Lessons

Bidirectional development fosters lasting synergies. Applying the TEI-ODD-Chaining concept to the software build process creates synergies that benefit multiple research projects. Each contribution of modifications or new features to the original source code reduces the maintenance burden in derivative projects, pooling expertise and resources across project boundaries.

Community building is essential for software sustainability. This approach can also lead to broader support for the software, laying the foundation for community building and ensuring the software's long-term viability. This is especially important for small, short-term digital humanities projects, but also for long-term projects without guaranteed follow-up funding.

Automation enables production-ready workflows. In Henze-Digital, the described process was used productively for over eighteen months and proved to be practical and robust, particularly for managing updates. Compilation and deployment of the research software as a web resource could be fully automated via CI/CD in the university's GitLab instance. This makes the workflow suitable even for smaller research projects. It is also possible to compile multiple software versions in parallel, as the base source code and project-specific modifications can be flexibly combined. As the repositories containing this infrastructure are not publicly accessible, please contact the author of this article if you have any further questions on this.

This approach demonstrates that well-designed strategies for software reuse can make digital humanities projects more efficient while contributing to the broader research software ecosystem.

Acknowledgements: Many thanks to the editorial team of Henze-Digital–Dr. Irmlind Capelle and Elena Minetti PhD–and our assistant–Sophie Stremel B. A.–for producing, enriching and cleaning so much data I could deal with. Further I'd like to thank the project leader–Prof. Dr. Antje Tumat–for applying the project. Last but not least, I am very grateful to Peter Stadler M. A. (WeGA) for his many suggestions and advisory discussions.

Bibliography

[BBC+24] T. Bachmann, A. Berndt, I. Capelle, K. Eich, A. Ferger, L. Frömmel, C. Hauck, S. Holst, E. Hufnagel, U. Konrad, N. Krämer-Reinhardt, A. Münzmay, S. Obert, D. Ried, F. Stickler, S. Stremel, O. Varwig, S. Waloschek. Postersession der 15. Edirom-Summer-School (Paderborn). Oct. 2024.
doi:10.5281/zenodo.14016119
https://doi.org/10.5281/zenodo.14016119

[Bur16] L. Burnard. ODD chaining for Beginner. Oct. 2016.
https://teic.github.io/PDF/howtoChain.pdf

[CKB+22] N. P. Chue Hong, D. S. Katz, M. Barker, A.-L. Lamprecht, C. Martinez, F. E. Psomopoulos, J. Harrow, L. J. Castro, M. Gruenpeter, P. A. Martinez, T. Honeyman, A. Struck, A. Lee, A. Loewe, B. van Werkhoven, C. Jones, D. Garijo, E. Plomp, F. Genova, H. Shanahan, J. Leng, M. Hellström, M. Sandström, M. Sinha, M. Kuzak, P. Herterich, Q. Zhang, S. Islam, S.-A. Sansone, T. Pollard, U. D. Atmojo, A. Williams, A. Czerniak, A. Niehues, A. C. Fouilloux, B. Desinghu, C. Goble, C. Richard, C. Gray, C. Erdmann, D. Nüst, D. Tartarini, E. Ranguelova, H. Anzt, I. Todorov, J. McNally, J. Moldon, J. Burnett, J. Garrido-Sánchez, K. Belhajjame, L. Sesink, L. Hwang, M. R. Tovani-Palone, M. D. Wilkinson, M. Servillat, M. Liffers, M. Fox, N. Miljković, N. Lynch, P. Martinez Lavanchy, S. Gesing, S. Stevens, S. Martinez Cuesta, S. Peroni, S. Soiland-Reyes, T. Bakker, T. Rabemanantsoa, V. Sochat, Y. Yehudi, R. F. WG. FAIR Principles for Research Software (FAIR4RS Principles). May 2022.
doi:10.15497/RDA00068
https://doi.org/10.15497/RDA00068

[CMR24a] I. Capelle, E. Minetti, D. Ried. HenDi-Data (data package). Nov. 2024.
doi:10.5281/zenodo.14227711
https://doi.org/10.5281/zenodo.14227711

[CMR24b] I. Capelle, E. Minetti, D. Ried. HenDi-ODD. Nov. 2024.
doi:10.5281/zenodo.14227688
https://github.com/Henze-Digital/HenDi-ODD

[Con25a] T. Consortium. Getting Started with P5 ODDs. 2025.
https://tei-c.org/guidelines/customization/getting-started-with-p5-odds/

[Con25b] T. Consortium. *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. 2025.
https://tei-c.org/guidelines/customization/getting-started-with-p5-odds/

[Rie24] D. Ried. HenDi-WebApp. Nov. 2024.
doi:10.5281/zenodo.14227708
https://github.com/Henze-Digital/HenDi-WebApp

[Rie25] D. Ried. Reuse and bidirectional further development of research software. Mar. 2025. Slides from the presentation at deRSE-Conference on February 27, 2025.
doi:10.5281/zenodo.15006673

[Rö20] T. Röder. Die offene Editionswerkstatt: Carl Maria von Webers Briefe in der digitalen WeGA. *RIDE. A Review Journal for Scholarly Digital Editions and Resources*, June 2020.
doi:10.18716/ride.a.12.4

[SNS$^+$25] P. Stadler, A. M. Neubert, S. Schreiter, S. Obert, D. Ried. WeGA-ODD. Apr. 2025.
doi:10.5281/zenodo.15212540
https://github.com/Edirom/WeGA-ODD

[SSZ$^+$25] P. Stadler, J. Schmidt, X. Zheng, D. Ried, K. Richts, S. Schreiter, C. Jakob. WeGA-WebApp. Apr. 2025.
doi:10.5281/zenodo.15212450
https://github.com/Edirom/WeGA-WebApp