International Journal of Digital Humanities (2024) 6:9–21
https://doi.org/10.1007/s42803-023-00076-9

ORIGINAL PAPER

Committing to reproducibility and explainability: using Git as a research journal

Samuel J. Huskey[1]

Received: 28 February 2023 / Accepted: 27 October 2023 / Published online: 19 Decemeber 2023

Abstract
Traditional critical editions in print offer a model for creating a culture of reproducible and explainable research in Digital Humanities. In addition to explaining their editorial methods and practices in a preface, editors provide a critical apparatus to allow readers to evaluate the evidence and decide for themselves whether the editor's judgment is sound. This paper will argue for viewing distributed, version-controlled data repositories (e.g., Git) in the same way, to empower users to follow the choices and methods of DH researchers and to reproduce their work. Using examples from a current project, this paper will propose some guidelines for using Git to promote reproducibility and explainability in the publication of DH research of all kinds.

Keywords Git · Version control · Digital humanities · Reproducibility · Explainability

1 Introduction

Over the centuries, textual scholars have developed an efficient and effective system for summarizing how and where the sources for texts differ from one another. That information traditionally accompanies the text of a critical edition in the *apparatus criticus*, or critical apparatus, the goal being to give readers the opportunity to evaluate the evidence for themselves and to provide support for the edition's argument. In other words, the aims of a critical apparatus are explainability and reproducibility. But one's first encounter with a critical apparatus can be anything but understandable, owing to the symbols and other typographical conventions unique to this system of scholarly

✉ Samuel J. Huskey
huskey@ou.edu

[1] Department of Classics and Letters, University of Oklahoma, 650 Parrington Oval, CARN 100, Norman, OK 73019, USA

communication, which is why readers must look to an edition's preface or a separate commentary for help.

In the effort to promote explainability and reproducibility in Digital Humanities (DH) research, we would do well to bear in mind the tension between the terse form of the critical apparatus and the narrative style of commentaries and notes on the text. Both have their place in explaining an editor's work and in aiding individual readers to evaluate the evidence for themselves. Whether we are working on digital editions, information visualizations, GIS mapping, natural language processing, or artificial intelligence and machine learning applications, the simple act of keeping a research journal would go a long way to making our work explainable and understandable to others. Because so much of that work occurs in the open on the internet, it is in the best interest of DH researchers to "aim to be at the forefront of documenting and sharing [their] processes and successes and failures in negotiating knowledge creation between very different epistemic cultures" (Edmond & Lehmann, 2021 ii105).

Elsewhere in this issue, Sarah Middle discusses the need for better documentation of DH research in general. This article complements hers by describing how the features of Git, a tool already in wide use in DH projects, could be leveraged in pursuit of that goal.

My observation of DH practice over the years is that projects make use of Git primarily to share their data. Presentations about DH research often include a reference to the project's data repository or repositories on a site such as GitHub (https://github.com/) or GitLab (https://gitlab.com/). So common are these references, in fact, that their absence, although not enough to impugn a scholar's DH *bona fides*, is noteworthy. But making data available in a Git repository is one thing; making the repository useful to other researchers is something else entirely.

In what follows, I advocate for the purposeful usage of Git as a tool not just for version control, but also for documenting DH research. Specifically, I call for widespread adoption of the practices of semantic commits and semantic versioning in DH projects, with a view to building intentional and meaningful research journaling into the Git workflow.

2 Case study: the Library of Digital Latin Texts

The context for this discussion is *The Library of Digital Latin Texts* (LDLT, https://ldlt.digitallatin.org/) a component of the Digital Latin Library (DLL, https://digitallatin.org/). The LDLT is a platform for publishing peer-reviewed, openly available, born-digital critical editions of Latin texts. LDLT editions are published under the authority of the University of Oklahoma, with sponsorship from three learned societies that manage the peer review process: The Society for Classical Studies, The Medieval Academy of America, and the Renaissance Society of America. The published products of the LDLT are not books or web pages, but Git repositories that contain the TEI XML file(s) for the edition and any other ancillary files the editor might want to include (e.g., transcriptions and/or collations of manuscripts; see Wittern (2013) for more on this concept). Using Git in this way supports the DLL's larger mission of

encouraging a shift toward considering critical editions as databases (Huskey, 2020). It has the added benefit of separating the rendering and visualization of the information in those databases as scholarly concerns in and of themselves (Witt, 2018). It also allows us to guarantee that a specific version of an edition has been peer-reviewed (Huskey & Witt, 2019).

Just as Git has influenced the way I approach digital critical editions, the nature of critical editions has influenced the way I think about Git. I will begin with a discussion of what Git has in common with the traditional humanities discipline of textual criticism. I will then turn to Git as a tool for documenting the decisions made in the development of an edition. I will conclude with suggestions for applying this Git-based approach to other kinds of DH projects.

2.1 Background: critical editions and Git

When searching for a text to encode as proof of concept for the LDLT, the DLL team settled on Caesar Giarratano's 1910 edition of Calpurnius Siculus' *Bucolica* (Giarratano, 1910) because it is relatively short, yet complex. As it happens, Giarratano's lifetime of work on the text of Calpurnius Siculus' *Bucolica*—he published four other editions of the text (Giarratano, 1924, 1939, 1943, 1951)—also offers a good basis for thinking about how to use Git in the development and publication of LDLT editions.

What interested the DLL team most about Giarratano's 1910 edition was its critical apparatus. A collection of variant readings, conjectures, abbreviations for manuscripts and previous editions, and other *arcana* traditionally printed in small type and occupying no more than a small fraction of the page, it takes up nearly half of every page in Giarratano's edition because it is positively bristling with variant readings from every source. It is, in fact, a positive critical apparatus, meaning that it reports not only variant readings and their sources, but also sources for the words printed in the main text of the edition (see West, 1973 87, note 14 on the use of "positive" and "negative" to describe a critical apparatus).

For example, *vaccae* ("the cows") is the last word in the fourth line of the first poem in the 1910 edition:

> cernis ut ecce pater quas tradidit, Ornyte, vaccae

The critical apparatus at that point in the text has the following:

> *vaccae* εχλβραηγ edd., *vacce* Ννφπκμ rp, *bacce* G, *vaccas* δ .

To the experienced reader, that is a clear and economical summary of how the sources differ at this point in the text. It communicates that eight manuscripts (denoted by the lower-case Greek letters) and most previous printed editions ("edd.") support Giarratano's decision to print *vaccae* as the lemma. There are also three alternative readings, in descending order of likelihood of being what Calpurnius Siculus wrote. The first of these alternatives (*vacce*) is found in six manuscripts (N, ν, φ, π, κ, μ)

and two early editions (r and p). The second and third alternatives (*bacce* and *vaccas*) have one witness each (G and δ, respectively).

The foregoing explanation of a single entry in the critical apparatus of an edition of a minor Latin poet may seem to have strayed rather far from the subject at hand, but it is relevant because it illustrates a fundamental concept of Git: Instead of tracking multiple copies of entire files in a project, Git tracks only what has changed. This is different, of course, from representing multiple documents with varying versions of the same text, as the apparatus of a critical edition does, but the principle is the same: Focus on the differences.

Giarratano's 1924 edition of Calpurnius Siculus' *Bucolica* gives us an opportunity to consider differences on a larger scale. The main text of the poem has not changed much, but the critical apparatus is entirely different. For one thing, instead of appearing at the bottom of every page, it appears at the back of the book as endnotes—much less convenient for scholarly readers. But that is not as significant as the change in the actual content. Looking at the first poem in this edition, we see that Giarratano still prints *vaccae* as the last word in line four, but the entry in the critical apparatus is noticeably different:

vacce ***N***, bacce ***G***, vaccas **ps**, vaccae *vel* vacce ***V***.

Most of the differences reflect Giarratano's decision to switch from a positive apparatus to a negative one, which reports only significant variant readings, leaving readers to conclude for themselves that any unmentioned manuscripts and previous editions must support the editor's choice to print *vaccae*.

There are other, more subtle, differences: Giarratano makes better use of the siglum ***V*** to represent the consensus of manuscripts in the second family. Moreover, when he needs to refer to individual manuscripts of that family, he uses lower-case Latin characters (e.g., ***p*** and ***s*** above) instead of the lower-case Greek characters from the 1910 edition. That is a potential source of confusion, since "r", for example, stands for the *editio Romana* of 1471 in the 1910 edition and for the manuscript Vratislaviensis Rehdigeranus I 4, 10 in the 1924 edition.

Unfortunately, the preface to the 1924 edition does not explain the reasons for these changes from the 1910 edition. In fact, the preface is mostly unchanged from one edition to the next, and Giarratano did not publish an explanation elsewhere.

2.1.1 If Git had been an option …

Although all the changes catalogued in the previous section were introduced at the same time in the 1924 edition, they are the result of several individual decisions to improve upon aspects of the initial version. Each time that Giarratano committed to implementing one of those decisions, he created a new version of the initial work. Eventually he had a version that he could tag as "final" and release into the world. Later he returned to the project and decided that a new version was needed, so he committed to doing the work, tagged it as a new version, and released it into the world

in 1939. He continued to work on the project, releasing two more versions of it in 1943 and 1951.

I selected the words in the previous paragraph from the vocabulary of Git to describe the steps in the publication of Giarratano's editions. "Initial version" is the basis of comparison for successive changes introduced to a project tracked by Git. "Commit" is one of the most common and important commands in Git. In the context of Git, "commit" is both a verb and a noun. As a verb, "to commit" simply means "to record changes to the repository"; a "commit", however, is a "single point in the Git history" (https://git-scm.com/docs/gitglossary#Documentation/gitglossary.txt-aiddefcommitacommit). In other words, a "commit" is a reference point for change or difference in a project; "to commit" is to add a reference point to a project's history. A "tag" is a label assigned to a specific commit to mark it as important. For example, it is common to tag a specific commit to mark a milestone in a project. Finally, a "release" is a bundle of all of a project's files in the state they were in at the time a specific commit was tagged.

Since Giarratano was working on a purely print paradigm for publishing his research, anyone wanting to compare his five editions of Calpurnius Siculus' *Bucolica* must find them in libraries or online and leaf through them manually to identify how they are different from each other. One could scan them, perform Optical Character Recognition, and write a program to compare the results, but it is doubtful that the resulting text of the critical apparatus would be reliable.

But if Giarratano were alive today and using Git to track his research project in a publicly available repository, anyone with access to the internet could view all five editions as releases, and they could use Git's commands to investigate precisely how they differ. Depending on how the repository is configured, they could also alert Giarratano to any problems with the edition through the issues queue. They could even contribute to the project by creating pull requests to incorporate their own work into the edition.

One simple, yet powerful, feature of Git has the potential to do more than any of the others to make Giarratano's research reproducible and explainable: the commit log, also known as a repository's history. The commit log is underused in many projects, yet it has great potential for bringing clarity to DH research projects for scholars, collaborators, peer reviewers, students, and others.

3 Using git as a research journal

I do not know whether Giarratano kept a journal of his work on the text of Calpurnius Siculus' *Bucolica*. If he did, I cannot imagine spending the time trying to track it down, if it still exists. Even if it does exist, and even if I could find it, there is no telling whether his notes to himself would mean anything to me. Even though he committed a significant portion of his life to the project, it takes extra commitment to keep a journal of one's research accurate and up to date. Myriad excuses offer themselves for putting off writing the next entry in a journal, particularly if you are keeping notes only for yourself.

Working in Git, however, makes shirking this responsibility difficult, since every commit requires a message, unless the user specifically overrides it with the `--allow-empty-message` option. That means that journaling is built into the workflow.

But that in itself does not solve the problem of haphazard record-keeping. In the absence of any project guidelines for using Git, contributors can make as few or as many commits as they please. And since there are few rules about the content of a commit message, it is possible, and unfortunately common, for contributors to do the bare minimum by writing generic messages (e.g., "Adding some files" or "Fixed a bug") without any further documentation, greatly reducing the usefulness of a project's log, as Randall Munroe humorously illustrates (https://xkcd.com/1296/). That represents a lost opportunity to cultivate practices that support reproducibility and explainability.

In the rest of this article, I argue for making better, more effective use of Git's commit log as a research journal. For examples, I will show how these guidelines would be implemented if the DLL were to update its pilot edition of Calpurnius Siculus' *Bucolica* to reflect the differences between Giarratano's 1910 and 1924 editions. The following proposals illustrate how I would use Git now, with the benefit of hindsight, not how I used Git when encoding the DLL's pilot edition.

3.1 Use clear and simple language

Elsewhere in this volume, Middle, building on Gibbs and Owens (2012), urges the use of clear and simple language in project documentation, without assuming anything about the technical abilities of end users. There is no denying that even with the availability of graphical user interfaces for Git, some will find it intractable. Researchers can mitigate that problem by providing the output of the Git log as a text file (e.g., `git log > gitlog.txt`) and by striving to make the content of commit messages as clear as possible. That means being thoughtful and deliberate about what a commit contains and how it is described.

3.2 Make one commit for each significant change

In English usage, the verb "commit" often occurs in connection with a significant change in status: a relationship ("to commit to one another"), finance ("to commit funds"), mental health ("to commit to treatment"), and law ("to commit a crime"). In Git, a `commit` should mark a single significant change in the project. A single significant change can affect one character or thousands of lines: it all depends on the nature of the change.

For example, one of the most noticeable differences between Giarratano's 1910 and 1924 editions is the switch from lower-case Greek letters to lower-case Latin letters to stand for eighteen of the manuscripts. A series of simple find-and-replace operations would seem to be called for here, but the process is complicated by the fact that Giarratano already used many of the same lower-case Latin letters in his 1910 edition to stand for previously published editions, so those must be changed

first to avoid serious confusion. In LDLT editions, moreover, sigla are also used as machine-readable identifiers (i.e., as values of the attribute xml:id) of manuscripts and editions (Huskey & Cayless, 2022 §7.2), so changes to those identifiers will break any references to them. Accordingly, the change of each siglum and its references is significant, so each one should be committed. The same is true of Giarratano's addition of two manuscripts as evidence for the 1924 edition.

Let us return to the cows in line 1.4 of Giarratano's 1910 edition, this time as encoded for the DLL's proof-of-concept edition. Here is the TEI XML for that line in its currently published state:

```
<app>
    <lem wit="#ε #χ #λ #β #ρ #α #η #γ"
    xml:id="lem–1.4–vaccae">
        vaccae
    </lem>
    <rdg wit="#N #ν #φ #π #ϰ #μ #r" source="#p"
    xml:id="rdg–1.4–vacce">
        vacce
    </rdg>
    <rdg wit="#G" xml:id="rdg–1.4–bacce">
        bacce
    </rdg>
    <rdg wit="#δ" xml:id="rdg–1.4–vaccas">
        vaccas
    </rdg>
</app>
```

The updates to the sigla and their references described above, not to mention the transformation of the critical apparatus from positive to negative, would require the refactoring of hundreds of lines of XML code.

It would be tedious and a waste of precious time to commit each one separately. It would be much more sensible to consider the updates to all of the apparatus entries for a single poem as a significant change worthy of a commit. Those interested in reviewing the individual changes could scroll through the output of the `git diff` command, which concisely displays the differences between any two commits.

In some places, however, Giarratano changes the text of the poems themselves, not just the way he reports the variant readings. Since he does not point out these changes anywhere in the preface to the 1924 edition, they can easily escape the notice of casual readers. Nevertheless, they alter the main text that scholars and major reference works cite. For example, *The Oxford Latin Dictionary* cites the 1924 edition, but the Packard Humanities Institute's *Classical Latin Texts* (https://latin.packhum.org/) uses the 1943 edition. In other words, the differences between Giarratano's editions make

a difference outside of their covers, so each one is significant and worthy of its own commit.

Some of the most noticeable differences between the 1910 and 1924 editions have to do with layout, but those differences affect the rendering of the edition's data, not the data themselves. Since the DLL treats the visualization of its editions as a separate scholarly concern, how to render the data in the format of the 1924 edition is a matter for a visualization project.

3.3 Write meaningful commit messages

The community of developers working on the Angular web development framework (https://angular.io/) has adopted the practice of adding a keyword or phrase at the beginning of each commit message and using a specific format for both short and long messages (https://angular.io/guide/doc-pr-prep#format-commit-messages-for-a-pull-request). These "semantic commits" make Git logs easier for humans and machines to use, particularly in situations where there are many human collaborators. The practice has spread to the wider development community. For example, the Conventional Commits specification (https://conventionalcommits.org/) recommends the following format for commit messages (elements in angle brackets are required; those in square brackets are optional):

```
<type>[optional scope]: <description>

[optional body]

[optional footer(s)]
```

It might make sense for some DH projects simply to adopt the Conventional Commits specification and recommendations without alteration, especially if the project is large and complex enough to benefit from automated continuous integration processes. But hewing to a standard designed for software engineers does not make good sense for many smaller projects, particularly those with few collaborators; and it is doubtful whether developing a separate standard with a list of `<type>` values for all humanities projects would be a good use of time and resources, given the range of knowledge domains that would need to be accommodated. Instead, simply following the basic pattern of the Conventional Commit standard (i.e., `<type>[scope]: <description>`) would greatly improve the usefulness of Git histories in individual DH projects of any type.

In the case of updating the sigla in Giarratano's edition, one might do the following:

```
change: siglum α to a for Ambrosianus O.74 sup.
```

Since the documentation for `git-commit` recommends keeping the subject within fifty characters, an (optional) description could be added to explain more complex changes:

```
change: siglum η to r for Vrat. Rehdig. I 4, 10

Note that r was previously used for the editio Romana,
whose siglum is now 'Romana'.
```

In the case of updating the critical apparatus for an entire poem, one could use the name of the TEI XML element `<app>` as the `<type>`:

```
app: Overhaul poem 1
```

For more complex changes, the various options allow more detail:

```
app(lem): Change 'umbras' to 'ulmos' at 2.5

Favoring the reading of P and V, which Jacoby
and Tolkhien support, over N and G.

MAJOR CHANGE TO EDITION TEXT
```

In this case, `app(lem)` identifies the scope of the change as a `<lem>` inside an `<app>` tag. A brief `<description>` summarizes the change. The optional `[body]` explains the reason for the change, and the optional `[footer]` alerts readers that this is a major change to the edition's text.

All of these details would be included in the output of `git log`, but, depending on the project, that output could extend to several screens. If someone wants to see only the major changes in the text, they could filter the output. For example, using the `--grep` option built into Git, one could enter the following on the command line:

```
$ git log --grep="MAJOR CHANGE"
```

The output of that command would be only the log entries with "MAJOR CHANGE".

3.4 Use annotated tags for milestones

Semantic Versioning, or SemVer (Preston-Werner, n.d.), another practice from the world of software development, complements this use of Conventional Commits.

Broyles (2022) recommends using a two-part system for the version numbers of digital editions: [major version].[minor version]. In that system, the major version number increases when an editor makes "a large number of changes systematically across the text that have a significant effect on its markup or on the way it is edited as a whole." The minor version number increases for any other change (e.g., corrections, minor adjustments to notes, etc.). Other DH projects might value the granularity of the tripartite pattern of the original SemVer specification: [major version].[minor version].[patch]. Regardless of a project's usage of SemVer, the version numbers should be used as the values for Git's `tag` command (https://git-scm.com/docs/git-tag) to label an accumulation of commits as a milestone in the project's development.

It is important to note here the importance of deciding when to mark a project milestone and how to label it. A simple way to do it in the Calpurnius Siculus project would be to use the publication years of Giarratano's editions as milestones, but that would imply that Giarratano's editions, not the text of Calpurnius Siculus' *Bucolica*, are the focus of the project. The adaptation of SemVer described by Broyles (2022) is a better approach for the main branch of the repository. If it is important to have the publication years as reference points, it would be trivial to name other branches after them. That would allow curious users to check out the `1910` branch or to use `git diff 1910..1924` to see the differences between the 1910 and 1924 editions. But in the main branch, the TEI XML encoding of the 1910 edition would be tagged as `1.0`. It would make sense to create another tag when all of the changes in the 1924 edition have been merged into the main branch. Certainly the change from a positive to a negative critical apparatus is a significant milestone, since it marks a change in editorial method. Accordingly, the 1924 edition in which that change was first implemented should be tagged as a major version change (i.e., 2.0). But the changes introduced in editions after 1924 were not as dramatic. It is true that the printed layout changed significantly between the 1924 and 1943 editions, but the content did not change as much. Giarratano did, however, change his mind about some readings in the main text of the edition, so that might be enough to increment the version to 3.0. But an argument could be made that the changes in the 1943 edition are minor and should be tagged as such (e.g., 2.1) It all depends on what the project's maintainers consider a "breaking change" (see Broyles, 2022 161).

Above all, it is important to annotate a project's tags so that users can make sense of the project's milestones (https://git-scm.com/book/en/v2/Git-Basics-Tagging). It is not necessary to provide a summary of all of the changes between tags, since users have access to the Git log and can view those changes however they wish. But knowing what the project's contributors consider to be important milestones can help users understand the project's history. For example, after merging the `1924` branch into the `main` branch of the edition of Calpurnius Siculus' *Bucolica*, I might issue the following command:

```
$ git tag –a 2.0 –m "Merge 1924 branch into main"
```

The simple message is enough to give a general sense of why the version number was elevated. Anyone looking for more information can look at the intervening commit messages or use `git diff` as described above.

4 Conclusion

The examples I have drawn on throughout this article come from only one branch of DH (i.e., digital philology), but the use of Git I have described here is well suited to DH projects of any kind, because it plays to the strengths of the best humanities scholarship, digital or otherwise: explanatory writing (through Conventional Commits) and systematic thinking (through Semantic Versioning and Git tags). It also buttresses efforts to improve the reproducibility, explainability, equitability, and sustainability of DH research.

For evidence that reproducibility is built into Git, I point to its `clone` command, used to make a complete copy of a project's data, including every commit message and annotated tag. Provided that the project's contributors have been intentional and disciplined about committing single significant changes, writing clear commit messages, and using annotated tags to label milestones, anyone who clones the project from an openly available repository will be able to follow the project's development, evaluate the contributions of different scholars on the project, interrogate the methodology, and attempt to reproduce the results.

That proviso also applies to explainability. There is no question that an explanation of any project should be published in a polished narrative form, but access to a project's Git history that has been kept as a step-by-step research log can yield different insights into the method and process behind that narrative description.

As for equitability and sustainability, the intentional use of Git can enhance those efforts, too. After all, Git is a free and open source software package that is available on all of the major computing platforms (Mac, Windows, Linux/Unix). Although it is most commonly invoked through a command line interface (CLI) such as the Terminal app, many graphical user interfaces or clients are available, many of them free of charge, for users who prefer a more visual experience. Git is also part of many integrated development environments (IDEs) popular with DH scholars, such as the free Visual Studio Code (https://code.visualstudio.com/). Remote storage and publication of Git repositories is also available at no charge at many research institutions and on the Open Science Framework (https://www.cos.io/products/osf). Commercial sites such as GitLab (https://gitlab.com/) and GitHub (https://github.com/, owned by Microsoft) also offer free hosting of Git repositories. Indeed, the repository for Git itself is on GitHub. For the sake of long-term availability and accessibility, researchers should consider hosting their own Git repositories on more than one of these services. And since Git works best with text-based files (e.g., XML, Markdown, CSV, etc.), it encourages scholars to work in formats with maximum accessibility. Indeed, Git's own files are easily viewed with any text editor. And since a Git repository includes both the project's data and its Git history, every clone of a repository, regardless of where it is hosted, is a potential backup.

To return to the example of textual criticism, the main purpose of a critical apparatus is to give readers access to the evidence for the text so that they can evaluate it for themselves. As opaque as it might seem at first sight, its purpose is to bring clarity to the argument advanced by the edition. It fulfills that purpose by recording and highlighting different versions of the text as printed in the edition. And yet, there are times when it would be helpful to see more details about the decisions represented in the critical apparatus; in those cases, scholars reach for commentaries and notes on the text. In a way, Git is a project's critical apparatus: a powerful tool for tracking different versions of files. But there are times when a narrative description would help us to understand the reasons for those changes, which is why thoughtful and systematic use of Git's log functionality should be an important part of the effort to document DH research, scholarship, and creative activity.

Author Contributions S.H. wrote the manuscript.

Funding Part of this work was supported by grants 21500706, 21400643, and 11200693 from the Andrew W. Mellon Foundation in support of the Digital Latin Library.

Availability of data and materials Not applicable.

Declarations

Ethical Approval Not applicable.

Competing interests The authors declare no competing interests.

References

Broyles, P. A. (2022). Electronic versioning and digital editions. In M. Vareschi & H. Wacha (Eds.), *Intermediate horizons* (pp. 147–166). Madison: University of Wisconsin Press. https://doi.org/10.2307/j.ctv2rh2cnh.12

Chacon, S., & Straub, B. (2014). *Pro Git*. Retrieved from https://git-scm.com/book/en/v2

Edmond, J., & Lehmann, J. (2021). Digital humanities, knowledge complexity, and the five 'aporias' of digital research. *Digital Scholarship in the Humanities, 36*(Supplement_2), ii95–ii108. https://doi.org/10.1093/llc/fqab031

Giarratano, C. (Ed.). (1910). *Calpurnii et nemesiani bucolica*. Neapoli: Apud Detken et Rocholl.

Giarratano, C. (Ed.). (1924). *Calpurnii et nemesiani bucolica*. Aug Taurinorum: In aedibus I.B. Paraviae.

Giarratano, C. (Ed.). (1939). *Calpurnii et nemesiani bucolica.* Aug Taurinorum: In aedibus I.B. Paraviae.

Giarratano, C. (Ed.). (1943). *Calpurnii et nemesiani bucolica.* Aug Taurinorum: In aedibus I.B. Paraviae.

Giarratano, C. (Ed.). (1951). *Calpurnii et nemesiani bucolica.* Aug Taurinorum: In aedibus I.B. Paraviae.

Gibbs, F., & Owens, T. (2012). Building better digital humanities tools: Toward broader audiences and user-centered designs. *Digital Humanities Quarterly, 6*. Retrieved from http://www.digitalhumanities.org/dhq/vol/6/2/000136/000136.html

Huskey, S. J., & Cayless, H. (2022). *Guidelines for Encoding Critical Editions for the Library of Digital Latin Texts.* Retrieved from https://digitallatin.github.io/guidelines/LDLT-Guidelines.html

Huskey, S. J., & Witt, J. C. (2019). Decoupling quality control and publication: The Digital Latin library and the traveling imprimatur. *Digital Humanities Quarterly, 13*(4). Retrived from http://digitalhumanities.org/dhq/vol/13/4/000438/000438.html

Huskey, S. J. (2020). Scholarly digital editions: A wise investment for scholars and institutions. In S. Chronopoulos, F. K. Maier, & A. Novokhatko (Eds.), *Digitale altertumswissenschaften: Thesen und debatten zu methoden und anwendungen* (pp. 43–54). Heidelberg: Propylaeum.

Preston-Werner, T. (n.d). *Semantic Versioning 2.0.0.* Retrieved 2023-02-28, from https://semver.org/

West, M. L. (1973). *Textual Criticism and Editorial Technique: Applicable to Greek and Latin Texts*. Stuttgart: Walter de Gruyter.

Witt, J. C. (2018). Digital scholarly editions and API consuming applications. In R. Bleier, M. Bürgermeister, H. W. Klug, F. Neuber, & G. Schneider (Eds.), *Digital scholarly editions as interfaces* (pp. 219–247). Norderstedt: BoD.

Wittern, C. (2013). Beyond TEI: Returning the text to the reader. *Journal of the Text Encoding Initiative, 4*. https://doi.org/10.4000/jtei.691