Developing Training and Education Resources for Research Software Engineering in DH

Julia Damerow[1], David Ragnar Nelson[2], and Jose Hernandez[3]

[1] School of Complex Adaptive Systems, Arizona State University, Tempe, AZ, USA
[2] Center for Digital Scholarship, American Philosophical Society, Philadelphia, PA, USA
[3] Research Computing Center, Florida State University, Tallahassee, FL, USA

Abstract

This paper outlines an in-progress educational resource created by the DHTech Education and Training Working Group. The preliminary resource seeks to fill a gap in the current training materials available to people who work on the technical side of Digital Humanities projects. It will provide resources on high-level core concepts of software development that contextualize, rather than duplicate, existing resources. The contribution outlines the intended workflow for producing this resource and details the anticipated challenges and barriers to implementation.

Keywords: pedagogy, training, digital humanities, research software engineering

1 Introduction

In 2025, DHTech[1] ran a survey to better understand the background and experiences of people working on technical aspects of Digital Humanities (DH) projects. According to this survey, 39% of respondents received their software development training through self-directed learning.[2] Additionally, about 70% reported that they are at least in part self-taught. There is nothing wrong with being a self-taught programmer, and the majority of software developers will be self-taught to an extent, whether to become familiar with a new framework or learning a new programming language. Small scale experiments and experiences suggest, however, that a formal software engineering or computer science degree provides a deeper understanding of computers and software development concepts [1, 6]. For research software, this lack of fundamental knowledge might contribute to some of the difficulties researchers face when writing code. Wiese et al. found that 70% of respondents to their survey of scientific R developers deal with technical difficulties such as software design and maintenance [13]. This is supported by a survey of scientific R developers conducted by Pinto et al., which found that 52% of respondents indicated they had only a novice understanding or no understanding at all of software design [11]. While systematic studies of software development skills in the humanities are lacking, Bleeker et al. found in a survey on "code literacy" in DH that around two-thirds of respondents reported having no, beginner, or intermediate levels of code literacy [3].

Given this need for self-directed learning, there is a demand for training resources for software development. In general, there are two types of resources: tool- or technique-specific resources and (research) software engineering specific resources. The first type typically focuses on a specific

Julia Damerow, David Ragnar Nelson, and Jose Hernandez. "Developing Training and Education Resources for Research Software Engineering in DH." In: *Digital Humanities Tech Symposium 2025*, ed. by Julia Damerow and Rebecca Sutton Koeser. Vol. 2. Anthology of Computers and the Humanities. 2025, 13–18. https://doi.org/10.63744/X9dgVaDTvJHY.

[1] `https://dh-tech.github.io/`

[2] The survey results will be published in the coming months. A similar survey run in 2020 provided similar results with 37% of all respondents reporting to be self-taught. See `https://dh-tech.github.io/survey-results-2020/`.

problem statement and how to use a certain tool or technique to address it; such as the lessons published in the Programming Historian[3] [7, 12]. Lessons in these resources teach, for example, how to create choropleth maps with Python, how to work with dates in R, or how to analyze newspapers with R. The second type of resource focuses on general software engineering topics and is not specifically targeting DH practitioners. The materials developed as part of the INTERSECT project[4] are one example, as well as the Code Refinery project,[5] the Carpentries,[6] Better Scientific Software,[7] and the US-RSE Education & Training Seminar Series.[8] Topics include how to design software, how to use Git and collaborate with others using Git and GitHub, and how to document software.

But there are disciplinary differences that affect the usefulness of these resources for software developers in the humanities. This was confirmed by the survey conducted by DHTech in 2025 which found that over 70% of respondents worked on web applications, projects, or digital editions, and 60% used JavaScript. In contrast, in the RSE International Survey 2022 only 19% indicated that they used JavaScript `hettrick2022`, suggesting that web application development is not the main task of many developers of research software outside of the humanities.[9] In addition, code and software are often not seen as primary research outputs in the humanities, which can be a barrier to starting conversations about best practices [14]. While the authors are not aware of any survey or study about the differences in required skills between DH developers and developers working in other research domains, personal experience and anecdotal evidence suggest that there are indeed variations. Although all of the resources listed above are highly valuable and widely used, there is a lack of materials that contextualize and bring together the separate topics and tutorials specifically with DH practitioners in mind. The Education & Training Working Group of DHTech[10] aims to fill this gap by offering lessons that link to existing materials where possible and contextualizing them as needed to provide a domain-specific perspective.

The goal of these lessons is to clarify that developing software is a multifaceted and complex process, with barriers that are unique to the digital humanities. Any large software development project can and should be broken down into smaller components; however, knowing how they work together in a maintainable and efficient way can be challenging. Therefore, we must know how to break down a problem into its constituent steps, carry out each step, and verify its success. For example, in a complex web application, the integration of different components, such as maintaining a database, designing a frontend, and handling a well-optimized and indexed search engine, requires sophisticated knowledge of software development. The digital humanities has an unfortunate history of projects that have disappeared from the web due to the creators' inability to maintain or migrate the project as its complexity has grown.[11]

In addition to the complexity of web applications, inexperienced programmers can introduce bugs or errors into the software. There are well-known examples in the hard sciences of bugs in research software leading to incorrect results [8]. While no such cases are known yet in DH, there is a real possibility for errors and bugs in our field as well. Even in a web application, which may

[3] `https://programminghistorian.org/`
[4] `https://intersect-training.org/`
[5] `https://coderefinery.org/`
[6] `https://carpentries.org/`
[7] `https://bssw.io/`
[8] `https://us-rse.org/wg/education_training/seminar_series/`
[9] The RSE International Survey 2022 did not ask any questions specifically geared towards the type of tasks developers of research software worked on. Hence, JavaScript is the strongest indicator regarding the question whether the respondent participated in web application development.
[10] `https://dh-tech.github.io/wg-education-training/`
[11] See, for instance, Quinn Dombrowski's reflections on failing to accomplish complex Drupal migrations: `https://quinndombrowski.com/blog/2019/11/08/sorry-all-drupal-reflections-3rd-anniversary-drupal-humanists/`.

not produce hard, verifiable results, bugs or errors in the software, such as an improperly designed search interface, could mislead end users and introduce errors into research. In addition, such errors matter even for researchers who never intend to release their code to be reused or as a web application, as they could lead to inaccurate or faulty results and conclusions.

2 There is "Work" in "Working Group"

The DHTech Education and Training Working Group was founded after discussions about training and education gaps of DH developers at the DH2023 conference. Its mission is two-fold: contextualize and bring together existing resources, as well as creating missing resources specifically aimed at developers of DH software. Neither of these tasks are trivial. Gathering existing resources and contextualizing them comes with the inherent risk of linking to outdated resources; outdated both in terms of content but also resources that are no longer available. While this is not an unsolvable problem, it does require an ongoing effort to mark outdated content and potentially even remove broken links from the materials. Additionally, while less work than composing new materials from scratch, it also requires time to find existing resources, vet them, and create a coherent story around them. Creating new resources comes with similar challenges, but the effort required for writing and reviewing materials is significantly higher.

The working group has created a website with lessons addressing the above mission. The group consists at this point of four volunteer editors that decide on the content of the website as well as the processes in use. So far, four lessons have been created; of these, two are published and two are in draft status. A published lesson is considered complete when it has been reviewed by someone other than the authors and has received a DOI to make it easily citable. A draft lesson is still a work in progress. Additionally, the website implements the concept of "pathways" that guide a reader through multiple lessons that thematically belong together. While this feature is still under development, it provides a powerful tool to provide different entry points and suggested learning materials based on the experiences of the reader. For instance, a lesson on Git and GitHub might be part of a pathway addressing novice programmers and a pathway addressing readers seeking materials for web development, allowing for the repurposing of materials without duplicating lessons.

3 The Technical Details

The website of the working group is created using the static site generator Hugo.[12] Hugo is open source and written in Go.[13] The website code is hosted in a GitHub repository[14] in the DHTech GitHub organization[15] and published using GitHub Pages.[16] There are several benefits of using GitHub's infrastructure, including free website hosting, ease of website updates (once changes to the content are made, a GitHub Action automatically deploys the new version), and potential future GitHub Actions to automate parts of the publication workflow. Additionally, GitHub Issues lend themselves to tracking the planning and maintenance of lessons as well as general website updates.

DOIs are generated through Zenodo.[17] Reviewed and accepted lessons are uploaded as PDFs into a Zenodo collection[18] and receive a DOI. This DOI is then added to the lesson on the working group website, along with the finished PDF file. The benefits of depositing finished lessons in Zenodo is that even if the working group website becomes unavailable in the future, the lessons

[12] `https://gohugo.io/`
[13] `https://go.dev/`
[14] `https://github.com/dh-tech/wg-education-training`
[15] `https://github.com/dh-tech`
[16] `https://pages.github.com/`
[17] `https://zenodo.org/`
[18] `https://zenodo.org/communities/dhtech-education`

will still be available.[19] Additionally, Zenodo allows versioning of lessons if they are updated in the future.

PDFs of lessons are created using Pandoc[20] and LaTeX. The workflow requires a lesson to be available as a Markdown file, which is the default format for Hugo pages. The benefit of this process is that the PDF generation could be done automatically using GitHub Actions. Setting up such a GitHub Action is planned as a future work element of the repository.

In addition, the working group anticipates that the next step towards sustainability and the incorporation of existing outside materials is to add a URL checker in a GitHub action, and retire technically outdated material in the lessons as modeled by the Programming Historian.[21]

3.1 Challenges

In the absence of any formal humanities institution for education and training in software development, the project is hindered at the moment by being a purely volunteer effort. Creating content takes time, which means progress is slow. However, with the infrastructure and processes now in place, adding new content is becoming easier. The ultimate goal is to create a community around this project to create new lessons and maintain existing ones. This requires, however, that there is enough incentive for volunteers to contribute which is why the group provides a DOI for published lessons: so they can be listed on a CV. In general, DH has long grappled with crediting practitioners for research outputs other than a traditional paper or article [9]; the inclusion of a DOI brings these training resources closer to a traditional research output. This is one approach to encourage contributions, and there may be others that could be explored in the future, such as further recognition for reviewers, or cataloging lessons for inclusion in other repositories of software engineering or digital humanities educational materials. It is the hope of this working group that once sufficient community has been built around the project, non-volunteer sources of labor may be feasible.

Maintenance of lessons is another challenge. Technologies become outdated and resources may become unreachable. The planned approach to combat this problem is three-fold. First, focusing more on concepts and processes rather than specific technologies may make lessons less likely to become obsolete. Second, one task of the editorial team of the working group is to periodically evaluate lessons for relevance and, if necessary, ask the author or another volunteer to update them. The future integration of an automated URL checker for URLs referenced in lessons would alleviate some of the workload, as well as, thirdly, the development of a lesson retirement policy, as mentioned above.

One question that many projects creating training resources must consider is how AI might change both the creation process and the content. Up to this point, the members of the working group see AI as a useful tool to be more efficient in developing software, but not as a replacement for well-designed, domain-specific training resources. Software development in general, and not just in DH, is actively grappling with the effects of generative AI on workflows, productivity, and training. Recent studies on the impact of AI on software development have been inconclusive. Two recent studies [4, 5] found generative AI had a positive effect on productivity in large software companies, while another [2] found a negative effect on productivity for open source software developers. In a recent study on the use of AI by scientists who write code, O'Brien found that scientific programmers without formal education in computer science used AI in place of official documentation, and that they often struggled to verify the output of AI-generated code [10]. The members of the working group are concerned about the effects these tools may have on errors and code maintainability, which makes robust training resources even more essential. In addition

[19] Zenodo is part of the CERN data center and promises that "your research is stored safely for the future in CERN's Data Centre for as long as CERN exists."

[20] `https://pandoc.org/`

[21] `https://programminghistorian.org/en/lesson-retirement-policy`

to using AI to write code, other advances in AI research could have effects on DH as well. For instance, the rise in the training or fine-tuning of LLM models for an ever-increasing number of humanities tasks, such as OCR correction,[22] will obligate the inclusion of AI software development and maintenance into the selection of lessons even if it is not used for our content creation and maintenance work.

4 Future Work

The most critical next task is to continue writing lessons and building out pathway lessons. This might include asking for feedback from the community and recruiting volunteers to work through the lessons. Furthermore, the editorial team will have to decide what topics to cover to ensure a coherent set of lessons.

Similarly important is the establishment of a community around the working group. Once a list of desired lesson topics exists, contributors can be recruited to write and review lesson materials. The long-term vision for the working group is a standing board of contributors and editors that support and maintain the website and materials. From a process perspective, while the infrastructure to review lessons is in place (utilizing GitHub pull requests), a more formal review process still needs to be established. This is particularly important if the number of concurrent lessons being developed continues to grow.

Lastly, the potential for connecting with existing efforts needs to be explored. At this moment, the working group is not yet at the point where this would be sensible, however, the long-term goal is to produce new lessons and materials in collaboration with other initiatives. This might include organizing collaboration workshops or similar events, which would also allow the group to continue filling the gap of materials for software developers in the humanities while reaching a wider group of practitioners in the digital humanities, making them aware of the hard work needed to keep our software working every day.

Acknowledgments

DHTech Education & Training Working Group consists of four members, only three of them participating in writing this article. Therefore, we want to give our heartfelt thanks to Fernanda Freire who is an integral part of this project.

References

[1] Bartram, Ammon. “Coding Boot Camp Grads Write Better Code”. https://www.infoworld.com/article/2250588/coding-boot-camp-grads-write-better-code.html. Accessed: 2025-07-30. 2016.

[2] Becker, Joel, Rush, Nate, Barnes, Elizabeth, and Rein, David. “Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity”. 2025. DOI: https://arxiv.org/abs/2507.09089. arXiv: 2507.09089 [cs.AI].

[3] Bleeker, Elli, Koolen, Marijn, Beelen, Kaspar, Melgar, Liliana, Zundert, Joris van, and Chambers, Sally. “A Game of Persistence, Self-doubt, and Curiosity: Surveying Code Literacy in Digital Humanities”. In: *DH Benelux Journal 4* (2022), pp. 1–28.

[22] https://aiforhumanists.com/

[4] Coutinho, Mariana, Marques, Lorena, Santos, Anderson, Dahia, Marcio, França, Cesar, and Souza Santos, Ronnie de. “The Role of Generative AI in Software Development Productivity: A Pilot Case Study”. In: *Proceedings of the 1st ACM International Conference on AI-Powered Software (AIware ’24)*. Porto de Galinhas, Brazil: ACM, 2024, pp. 1–8. DOI: 10.1145/3664646.3664773.

[5] Cui, Zheyuan, Demirer, Mert, Jaffe, Sonia, Musolff, Leon, Peng, Sida, and Salz, Tobias. “The Effects of Generative AI on High-Skilled Work: Evidence from Three Field Experiments with Software Developers”. http://dx.doi.org/10.2139/ssrn.4945566. Accessed: 2025-07-30. 2025.

[6] Kimball, Danika. “Self-Taught Developers vs. Developers With Degrees”. https://manilarecruitment.com/manila-recruitment-articles-advice/self-taught-developers-vs-developers-with-degrees-hiring-the-right-fit/. Accessed: 2025-07-30. 2023.

[7] Mullen, Lincoln A. “Computational Historical Thinking”. https://dh-r.lincolnmullen.com/. Accessed: 2025-07-30. ongoing.

[8] Neupane, Jayanti Bhandari, Neupane, Ram P., Luo, Yuheng, Yoshida, Wesley Y., Sun, Rui, and Williams, Philip G. “Characterization of Leptazolines A–D, Polar Oxazolines from the Cyanobacterium Leptolyngbya sp., Reveals a Glitch with the ‘Willoughby–Hoye’ Scripts for Calculating NMR Chemical Shifts”. In: *Organic Letters 21*, no. 20 (2019), pp. 8449–8453. DOI: 10.1021/acs.orglett.9b03216.

[9] Nowviskie, Bethany. “Where Credit Is Due: Preconditions for the Evaluation of Collaborative Digital Scholarship”. In: *Profession* (2011), pp. 169–181. DOI: 10.1632/prof.2011.2011.1.169. URL: http://www.jstor.org/stable/41714117.

[10] O’Brien, Gabrielle. “How Scientists Use Large Language Models to Program”. In: *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2025, pp. 1–16. DOI: 10.1145/3706598.3713668.

[11] Pinto, Gustavo, Wiese, Igor, and Dias, Luiz Felipe. “How Do Scientists Develop Scientific Software? An External Replication”. In: *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 2018, pp. 582–591. DOI: 10.1109/SANER.2018.8330263.

[12] Ryan, Yann. “A Short Guide to Historical Newspaper Data, Using R”. https://bookdown.org/yann_ryan/r-for-newspaper-data/. Accessed: 2025-07-30. 2021.

[13] Wiese, Igor, Polato, Ivanilton, and Pinto, Gustavo. “Naming the Pain in Developing Scientific Software”. In: *IEEE Software 37*, no. 4 (2020), pp. 75–82. DOI: 10.1109/MS.2019.2899838.

[14] Zundert, Joris van, Antonijevic, Smiljana, and Andrews, Tara. “Digital Technology and the Practices of Humanities Research”. In: Open Book Publishers, 2020. Chap. 6. ’Black Boxes’ and True Colour — A Rhetoric of Scholarly Code, pp. 123–162. DOI: 10.11647/OBP.0192.06.