**DHQ: Digital Humanities Quarterly**
2022
Volume 16 Number 2

# Relocating Complexity: The Programming Historian and Multilingual Static Site Generation

Matthew Lincoln <mlincoln_at_andrew_dot_cmu_dot_edu>, JSTOR Labs
Jennifer Isasi <jenniferbibat_at_gmail_dot_com>, Pennsylvania State University
Sarah Melton <sarah_dot_melton_at_bc_dot_edu>, Boston College
François Dominic Laramée <fdl_at_francoisdominiclaramee_dot_com>, University of Ottawa and Université de Montréal

## Abstract

In this case study, we show how the challenges of maintaining a sustainable static-site architecture for the *Programming Historian* are deeply intertwined with the logistical challenges of expanding the original project into a multilingual set of publications. In our desire to democratize access to knowledge, we constantly encounter situations where easing the complexity of one workflow requires increasing the complexity of another, in turn relocating complexity (and the labor it entails) between different members of our project team.

## Introduction[1]

Like any technologists, practitioners embracing minimal computing principles must grapple with the changing purposes, audiences, scales, and functionalities of the project they are supporting. In this case study, we explore the challenges of maintaining a sustainable static-site architecture during the multilingual expansion of *The Programming Historian (PH)*, an open-access publication of peer-reviewed tutorials on digital tools and workflows geared at humanities teachers and researchers. By elucidating the deep intertwining of the technical, logistical, and social challenges of moving from a monolingual to a multilingual publication, we hope to provide other digital project teams with a valuable perspective on the tradeoffs of static sites when the projects they support are rapidly growing in size and complexity. 1

*PH* started in 2008 as a series of open access tutorials on Python created by William J. Turkel and Alan MacEachern. The lessons were hosted by the Network in Canadian History & Environment (NiCHE) on a WordPress site [NiCHE 2009]. In 2012, the project expanded its editorial team and became an open-access, peer-reviewed scholarly journal of methodology for digital historians. In 2014, *PH* migrated away from WordPress in favor of a Jekyll-based static site hosted on GitHub Pages [Gibbs 2015]. The project team made this decision based on the benefits promised by many static-site generators: fewer worries about security upgrades and hosting expenses; precise customizability for our editorial workflow; and preservable, version-controlled, plain text files for all of our peer-reviewed lessons. Since this migration, we have never: 2

1. Run out of memory, processing power, or bandwidth;[2]
2. Worried about SQL injection or external hacking;
3. Lost data from accidental deletion or hardware failure; or
4. Been bound by the built-in data models provided by a platform such as Open Journal Systems or Janeway.

At the time, with only one team of editors managing a single publication workflow, this minimalistic approach seemed an excellent solution for both technical flexibility as well as sustainability. 3

But *PH* has grown radically since then. Prompted by a discussion about the use of *PH* in Colombia and Mexico and ad-hoc translations into Spanish, the *PH* editorial team decided to extend the journal to a second language [PH Issue 246]. 4

In late 2016, a group of three editors began work on *PH en español*, launching the site in March 2017. A year and a half later, a group of French-speaking editors did the same, launching *PH en français* in April 2019. A Portuguese editorial team launched *PH em português* in January 2021.

5 Throughout the process of onboarding new editors from multiple backgrounds and publishing lessons in several languages, both originals and translations, the team came to discuss ways to better reach our global audience and enhance language accessibility.[3] These goals have relied on the malleable Jekyll-based site to create a more accessible journal, but have also tested the limits of both the technology as well as our team's processes. We frame this work as an example of "critical making," defined by Matt Ratto as "connect[ing] two modes of engagement with the world that are often held separate — critical thinking, typically understood as conceptually and linguistically based, and physical 'making,' goal-based material work" [Ratto 2011, 253]. For *PH*, creating and maintaining a multilingual publication is an act of critical making, negotiating the complexities of language and the constraints of our infrastructure in both technology and labor. A simple question on additional metadata, a request for search capability, or a light-bulb moment when someone asks, "How are we going to note if someone translates a translated lesson?" requires sociotechnical expertise in different levels of our publishing workflows and technical infrastructure from a team of volunteer humanists.

6 In our desire to democratize access to knowledge, we constantly encounter situations where easing the complexity of one workflow requires increasing the complexity of another, in turn relocating complexity (and the labor it entails) to different members of our project team.

## Language Expansion

7 Nowhere is the shifting of complexity between team members more obvious (or problematic) than when the project expands by adding a publication in a new language. Linguistic expansion is part and parcel of *PH*'s core mission of opening access to learning across geographic and cultural boundaries [Sichani et al. 2019] [Crymble 2016]. However, new language teams face multiple challenges, both before and after they start publishing content. Here, we focus on the experiences of the French team, which was formed in early 2018. The Portuguese team, which was onboarded as we wrote this article, is likely to face similar issues, perhaps made more acute by the expansion of the project's infrastructure over the past three years. Any additional teams joining the project in the future will meet similar obstacles.

8 The first hurdle, which must be cleared before any publication in a new language can occur, consists of translating the project's interface, documentation, and accumulated wisdom. This requires that the new language team assimilate this knowledge, along with the technical know-how necessary to navigate the *PH*'s underlying technology. In the case of the French team, this process took the better part of a year. If, as Alex Gil wrote, one of the precepts of minimal computing is to restrict ourselves to "what we need," what new *PH* language teams need has proven considerable indeed [Gil 2015]. Of course, some of the work, such as the translation of menus and author guidelines, would have been necessary regardless of the publishing platform. However, much of the rest — including self-training on the somewhat arcane procedures (e.g., how to write markdown, how to commit a change in GitHub, how to format metadata) — that must be followed to publish lessons, as well as translating these procedures into the new publication's language for the benefit of future editors, is considerably more involved than it would be if the project used a standard Content Management System (CMS). Whereas most CMS software is built upon user interface concepts that have long been internalized, our procedure requires a fairly steep learning curve that often must be climbed several times before the procedure becomes comfortable. Some of the complexity of a CMS has thus been offloaded onto editors.

9 Language expansion also has subtle implications for project maintenance [Smithies et al. 2019]. When content needs to be added or modified, either in the site's common infrastructure or in a lesson that has been published in more than one language, two options are available: publish the update in all languages at the same time or update each language-specific site as new content becomes available. In the first case, the language team with the least available bandwidth constrains the speed of the update. In the second case, the project's content becomes desynchronized, and a backlog of tasks accumulates for one or more of the language teams. Either way, smaller teams experience asymmetrically heavier pressure. Since its inception, the French team has rarely had more than two fully active members at the same

time; the pressures of day-to-day maintenance thus fall on few people, which in turn slows down the production of new content. One consequence is that some translations have been held up for several months, for lack of an available editor to manage peer review. As the project expands to more languages, pressure will increase on these smaller teams to keep pace, and may disrupt the smooth running of the project.

A further source of complexity resides in the dual pipelines required to publish translated and original lessons in each language. For example, recruiting peer reviewers to examine translations of tutorials into French has proven challenging because, while the work is relatively straightforward, there are no clear norms within academia to reward it. (Where does one even list peer-reviewing a translation on their CV?) Thus, the French team's editors have had to perform more of this work than perhaps would be wise, further depleting the amount of bandwidth available for other tasks. 10

Finally, while language expansion provides *PH* with opportunities to further the cause of global outreach and help multiple digital humanities traditions cross-pollinate, it also increases labor complexity for the editors.[4] Our hope is that authors will eventually produce original content in their own languages for all of our publications. To further this cross-pollination, however, we will also need to recruit translators able to adapt source material written in multiple languages and reviewers able to assess this work. It seems inevitable that, as we advance in this direction, we will push against the limits of what is adaptable, which will lead our publications to diverge. Finding the right balance will present interesting challenges.[5] 11

## Code Complexity vs. Workflow Complexity

It should be clear by now that *PH* has an involved publication workflow. From our experience running such an intricate publication via Jekyll, we would argue that static sites cannot eliminate complexity from already-complex publications; they can only relocate it. 12

The database systems behind conventional content management systems (CMS) such as Wordpress or Drupal do more than store and serve data. They enforce crucial logical constraints such as type checking (*Is this date valid? Is this field an integer rather than a text value?*); uniqueness (*Does another page already have this URL? Does a tag with this title already exist?*); and foreign key relationships (*This article is written by two different authors — do they both have corresponding entries in the "authors" table?*). In the case of multilingual sites, some CMSs don't provide these options and, when they do, their functions are not necessarily straightforward or accessible. For example, one can only track progress of a translation if using the paid for WordPress Multilingual Plugin. With Drupal, the interface has to be translated manually on individual pages. And other options, such as .NET, require the user to be comfortable editing HTML and XML to work with RESX or resource files. 13

Static site generator systems such as Jekyll offer a pared-down version of data manipulation. For every article in *PH*, some data reside in the lesson file (the title; the abstract; the date of publication; the original review ticket URL; the names of the authors, editors, reviewers, and translators; content tags; and a reference key from translated lessons to the original lesson) while other data (author biographies) sit elsewhere. Without the querying features or the constraints supplied by an active database, the onus of maintaining metadata consistency in a straightforward static site falls to its editors. 14

At *PH* we sought technological opportunities to transfer some of this burden away from our editorial team and onto our codebase instead. Our technical team has poured labor into creating metadata validation systems that would have otherwise come built-in with a database-backed CMS. As our team, metadata richness, and pace of publication expanded, metadata errors increased in frequency. And with a larger team, it was less likely that any given editor would be completely familiar with how the data they entered was used by the site, making it more likely that multiple editors would need to be pulled into a troubleshooting conversation. 15

To ameliorate this, in May 2017 we wrote the first version of an internal plugin where we could define the kinds of data validation that a normal database would have provided, such as validating required fields, dates, and foreign key relationships. Anyone building the site on their own computer will now see informative error messages describing which fields are missing or which names can't be found in the data file. With this plugin, we also implemented dead link 16 checking to review all our pages for internal and external links that no longer resolve, alerting our editors to update them or to replace them with links from the Internet Archive's Wayback Machine if available [Lincoln 2017].

Yet running this plugin means the editor needs to be able to build the Jekyll site on their own computer. This is a requirement we gave up on long ago, relying on most editors to use GitHub's code editing interface and using GitHub Pages to actually build and deploy the site. To run this validation, we configured an external continuous integration service: a separate server that would test metadata validity every time an editor proposed a new change to the site and report back any errors in a relatively human-readable format to our editorial workflow system in GitHub's pull requests. As a result, our minimal publication was no longer all that minimal. 17

At each step of this process, we opted for increasing the amount of underlying complexity in the site template code, processing code, and in the number of back-end deployment steps. We did this to reduce the cognitive load on and demand for technical competencies from our editors. They now do not need to memorize all required data fields and do not need to be able to run Jekyll themselves but can just read the reports from our cloud-based testing system. 18

We have repeated this complexity relocation several times, particularly in automating the cross-linking of different pages of the same language as much as possible. Adding hard links between all translations of every single page or lesson would require constant vigilance as pages are updated and new translations published. Additionally, the number of links would grow exponentially as we bring on new language teams. To avoid asking editors to do this grueling work, we instead relied on the affordances of Jekyll and its templating language, Liquid, to construct virtually all the links between our lesson translations. 19

While this relocation saves a large amount of editor time, it significantly increases the complexity of our site's source code [Lincoln 2020]. And any increase in our code complexity means a concomitant increase, however slight, in the cognitive load and work of the technical team in charge of maintaining and extending *PH*'s online platform. This new code was complex enough to require its own dedicated tutorial page. And even with that documentation, changes to the way we render translation links is a delicate job indeed, requiring skilled oversight from our technical editors.[6] 20

Every choice to displace the labor away from our lesson editors onto our code base thus displaces further cognitive load onto our technical team in charge of maintaining every part of *PH*'s technical infrastructure. This raises serious questions about sustaining the ongoing labor of the project. It forces us to more carefully consider every new feature request that comes from our larger team, weighing the potential advantages against the time and difficulty needed to implement it. As our publications continue to expand, we have an opportunity to refocus the energy of our entire project team onto the essentials of our publications, concentrating on managing the inevitable but critical logistical complexity of multilingual translation, while forgoing new technical features that would merely be nice to have. 21

## Conclusion

Knowing what we know now, would the *PH* project team still choose to publish using a static site architecture? For all the challenges that we have laid out in this case study, we don't intend to warn against static sites altogether. Free from the task of monitoring the basics of a database-backed site, our technical team has instead been able to spend time on customizing our lessons' display and metadata to fit our exact needs and building out services such as publication metadata registration and DOIs, all while ensuring our publication record is carefully tracked and preserved in plain text. These benefits are real. But we do want to draw attention to the careful (and not-so-careful) choices we have made about how to distribute the labor of editing and maintaining our publication. 22

In response to the stress of increasing sociotechnical complexity, the *PH* project team has pursued social changes to ensure that the labor required by this complexity is understood by the entire team. Among these changes have been formalizing sub-teams of the *PH* to more clearly delineate roles and responsibilities; articulating a workflow and communication plan for managing bug fixes and new feature requests so that the technical experts on the project team are able to communicate their workload and reasons for pursuing (or not) suggested changes; and establishing team roles specifically for onboarding new editors and new language teams to help address the barriers to joining a new publication. 23

There are no simple social or technical solutions for publications as complex as *PH*, and the choice of a static architecture still presents tradeoffs with real consequences for the sustainability of any project. We hope this case study highlights the nature of those choices, and helps other digital publication teams consider where (and on whom) the burden of managing complexity falls in their own projects.

## Notes

[1] Matthew Lincoln was the technical lead for *PH* from 2017–2020. Jennifer Isasi has been a member of the technical team and *PH en español* editorial team since 2018. François Dominic Laramée was a member of the *PH en francais* editorial team 2018–2020. Sarah Melton has been the managing editor of the English editorial team of *PH* since 2019.

[2] Based on statistics provided by Cloudflare, we served 315GB of data in the month of May 2020 alone, which would have incurred significant annual costs if we were running *PH* through our own dedicated server rather than relying on GitHub's (presently) free content hosting. We wish to clearly acknowledge this dependency on an outside company whose ties to the United States' Immigrations and Customs Enforcement give us cause for concern [Lecher 2019]. We have the technical capability to shift our source code, site hosting, and issue tracking to another platform if need be without losing data, but we balance this possibility against the renewed learning curve it would demand from our editors — a critical concern for us, as we discuss below.

[3] Presently, working on *PH* infrastructure requires that editors know English to navigate editorial team discussions, translation workflows, and internal documentation. This barrier is lower for peer-reviewers of originals, but it is still a challenge for reviewers of translations.

[4] For a discussion of the global dimensions of digital humanities, see Alex Gil and Élika Ortega. "Global Outlooks in Digital Humanities," *Doing Digital Humanities: Practice, Training, Research*, edited by Constance Crompton, Richard J. Lane, and Ray Siemens (NY: Routledge, 2016), pp. 22-34.

[5] To learn more about the balance between editorial and translation work that puts the needs of the audience first while shaping the editorial workflow, see [Isasi and Rojas Castro 2021] on types of translation in PH.

[6] See, for example, in this ticket one can read the amount of back-and-forth required for a deceptively simple change in the way that we display links from one lesson to its other language versions.

## Works Cited

"Multilingual Guide," *Drupal Documentation*. Available from: https://www.drupal.org/docs/multilingual-guide. Date accessed: August 25, 2021.

"Localization in .NET," *Microsoft Docs*. Available from: https://docs.microsoft.com/en-us/dotnet/core/extensions/localization. Date accessed: August 25, 2021.

**Crymble 2016** Crymble, Adam. "Identifying and Removing Gender Barriers in Open Learning Communities: *The Programming Historian,*" *Blended Learning in Practice* (2016): 49-60, https://researchprofiles.herts.ac.uk/portal/files/10476604/Blip_2016_Autumn_2016_Final_Autumn_2016.pdf.

**Gibbs 2015** Gibbs, Fred. "Editorial Sustainability and Open Peer Review at *Programming Historian,*" *DH Commons*, 1 (2015). https://web.archive.org/web/20180713014622/http://dhcommons.org/journal/issue-1/editorial-sustainability-and-open-peer-review-programming-historian.

**Gil 2015** Gil, Alex. "The User, the Learner and the Machines We Make," *Minimal Computing: A Working Group of GO::DH*, May 21, 2015.https://go-dh.github.io/mincomp/thoughts/2015/05/21/user-vs-learner/.

**Isasi and Rojas Castro 2021** Isasi, Jennifer and Antonio Rojas Castro. "¿Sin equivalencia? Una reflexión sobre la traducción al español de recursos educativos abiertos." *Hispania* vol. 104.4 (December 2021). https://doi.org/10.1353/hpn.2021.0130.

**Lecher 2019** Lecher, Colin. "GitHub Will Keep Selling Software to ICE, Leaked Email Says." *The Verge*, October 9, 2019. https://www.theverge.com/2019/10/9/20906213/github-ice-microsoft-software-email-contract-immigration-nonprofit-donation.

**Lincoln 2017** Lincoln, Matthew. "Infrastructure for Collaboration: Catching Dead Links And Errors," *The Programming Historian Blog*, July 31, 2017. https://programminghistorian.org/posts/infrastructure-at-ph.

**Lincoln 2020** Lincoln, Matthew. “Multilingual Jekyll: How *The Programming Historian* Does That,” (personal blog) March 1, 2020. https://matthewlincoln.net/2020/03/01/multilingual-jekyll.html.

**NiCHE 2009** “The Programming Historian,” *Network in Canadian History & Environment*. Accessed April 10, 2022. http://web.archive.org/web/20091211210942/http://niche-canada.org/programming-historian.

**PH Issue 246** “Spanish translation - Issue #246,” *Programming Historian GitHub Repository*. Accessed April 10, 2022. https://github.com/programminghistorian/jekyll/issues/246.

**Ratto 2011** Ratto, Matt. “Critical Making: Conceptual and Material Studies in Technology and Social Life,” *The Information Society: An International Journal*, vol. 27.4 (2011): 252-260. https://doi.org/10.1080/01972243.2011.583819.

**Sichani et al. 2019** Sichani, Anna-Maria, James Baker, Maria José Afanador Llach and Brandon Walsh. “Diversity and Inclusion in Digital Scholarship and Pedagogy: The Case of The Programming Historian”, *Insights* (2019). https://doi.org/10.1629/uksg.465.

**Smithies et al. 2019** Smithies, James, Carina Westling, Anna-Maria Sichani, Pam Mellen and Arianna Ciula. “Managing 100 Digital Humanities Projects: Digital Scholarship & Archiving in King’s Digital Lab,” *Digital Humanities Quarterly*, vol. 13.1 (2019). https://www.digitalhumanities.org/dhq/vol/13/1/000411/000411.html.