The MIT Press

MIND THE GAP
A Landscape Analysis of Open Source Publishing Tools and Platforms

John W Maxwell, Erik Hanson, Leena Desai,
Carmen Tiampo, Kim O'Donnell, Avvai Ketheeswaran,
Melody Sun, Emma Walter, Ellen Michelle
Canadian Institute for Studies in Publishing, Simon Fraser University

mindthegap.pubpub.org | DOI 10.21428/6bc8b38c.2e2f6c3f

Table of Contents

Introduction

Over the past two decades, a new breed of publishing infrastructure has emerged via open-source software. Where publishing toolchains had previously been almost entirely populated by proprietary and often bespoke software systems, we now see a proliferation of open-source projects available for adoption and integration—on a different economic and operational footing. Many such projects have been designed and developed by a single institution to suit its own particular needs, but the terms of open-source software licensing, deployment, and indeed governance mean these systems are also readily available to other institutions. At a more ambitious level, they may even form a layer of *community infrastructure* that rivals—or at least provides a functional alternative—to the commercial infrastructure run by a small number of for-profit entities.

That such a proliferation of open-source projects now exists is a boon, but the landscape is noisy and difficult to understand as a whole. There is no guidebook or map to this landscape—a problem the present report seeks to address. MIT Press, in its 2018 application to the Andrew W Mellon Foundation, identified the need for a "comprehensive and critical analysis of OS publishing systems in active use" that "could prove to be durable alternatives to complex and costly proprietary services." The present report is the result of that research and analysis.

Our hope is that this report will provide the university press community and other mission-focused enterprises with both an overview of the open-source landscape as well as profiles of a good number of these projects individually. Our intention is to shed light on the development and deployment of open source publishing technologies in order to aid institutions' and individuals' decision-making and project planning.

There is enormous value in the collection of open-source projects surveyed here in terms of raw functionality as well as in the ways that prototyping and the evolution of design materially change the ways in which we think about publishing and scholarly communications. At a more detailed level, this report seeks to encourage the adoption and continued development of these platforms, but also to encourage the development of the community and market environment that surrounds these efforts.

As such, while this report provides a catalogue of individual open-source publishing tools (see Part II), it also examines the *ecosystem* in which these tools and projects exist. If publishers are to develop or find robust, cost-beneficial alternatives to commercially obtainable services and systems, it will not be simply because free tools exist; rather, it will depend greatly on community practices and the integration of various tools into a broader interoperable context. The idea of *community infrastructure* is not just a collection of bits of technology, but a system in which these components can be mobilized to serve larger goals.

Method

Our team at the Canadian Institute for Studies in Publishing at Simon Fraser University began this landscape analysis in summer 2018 by assembling a master list of projects. We were helped by a number of existing lists of projects and initiatives that had been compiled by various colleagues (notably from Terry Ehling, Kevin Hawkins, Peter Suber, Adam Hyde, the Radical OA Collective, and JROST). From this beginning we needed to filter the list—in the first place for projects that fit the scope of our project: available, documented open-source software relevant to scholarly publishing. Second, we sought to identify projects that were 'still alive'—that is, with evidence of active development. This latter criteria is somewhat difficult, because the Internet tends to flatten history—things from decades past appear alongside much more recent contributions. There is no telltale yellowing of web pages. The sifting of old, dormant projects with vibrant-sounding websites from active projects that people still care about took considerable time and attention.

By mid-winter we had assembled a list of approximately 85 projects that appeared to be active. In the early months of 2019, we did a deeper dive into these projects, locating their code repositories (almost always on Github, with a handful using Gitlab instead), tracking down details of personnel, funding, and especially, evidence of partners and collaborators. We travelled to conferences, asked questions over email, and conducted dozens of Skype, Zoom, and even old-fashioned telephone calls. By April, we had winnowed the list down to approximately 50 projects. Some we dropped because it became apparent the projects were in fact dormant; some because we decided they were out of scope for our project; some we realized were part of larger assemblages. We believe that the current list represents the field well. That said, this is a dynamic space, and our cataloguing is a snapshot of a moment in time. By the time you read this, some of the details will already be out of date.

The present report is in two parts. Part I provides some high-level analysis of the landscape and the projects within it. In the first section of Part I, we discuss the scope of this report, define some working terms, and set the larger context for the projects we survey. Next, we attempt to break down the field along a number of axes, providing some provisional categorization of the projects—from their goals and organizational structures to specific technological approaches. In the final section of Part I, we explore the prospects for sustainability, collaboration, and interoperability within the current landscape, and suggest some opportunities for new initiatives based on this analysis.

Part II of this report is a catalogue of the projects themselves. For each open-source project, we provide a summary description plus details on the host organization, the project's principal investigator or leadership, funders, partners (both strategic and development), date of original release, and current version. We also include some basic data drawn from the Github/Gitlab repositories for these projects, including development language, license, number of contributors. Our initial ambitions to conduct a full "Github audit" proved not feasible, because most of the projects surveyed are small, and with varying project management and organizational approaches—as such the metrics Github provides on activity are not useful in a comparative context.

Key themes

While the primary focus of our research, and of this report, is software and software development—*functionality, code, developers, partners, and funders*—the themes we have kept in mind throughout have to do with *sustainability, scale, collaboration, and ecosystem integration*. Through all of our research, and our investigations of dozens of projects, the question in the back of our minds is always *who will care about these projects?* Their project leads and PIs of course care, but beyond the inner circle of active agency… who else will care enough to fund, contribute, promote, use, and ultimately further the useful life of these projects? What are the values and mechanisms that cause people—especially external stakeholders—to care enough about these projects to keep them alive, and even thriving, going forward?

There are a great many projects here. From a distance, if one squints, some of these projects seem to cover the same ground, to provide the same functionality. Looking closer, however, the overlap is less obvious; indeed, it becomes clearer that each project is designed or evolved to fit a particular niche, to solve a specifically formulated problem. The result is a complex, multi-faceted landscape that defies easy categorization, let alone identifying "best-of-breed" applications from among several contenders.

When we were conceiving of this landscape report, we talked of its role as a *gap analysis,* where amid the many development initiatives, we might identify an underserved area where new development would be most valuable. But this has not proved to be an obvious outcome of this study; rather, there is a lot of functionality out there—a lot of code, a lot of thinking, and a host of very context-specific approaches to basic publishing functions.

What there isn't much of is *coordination* between these approaches. There isn't a good deal of interoperability between many of these projects, and there is in places definite overlap in goals (if not in specific strategies). We've noted that there aren't obvious incentives for collaboration between projects. As such, if there is a 'gap' that can be identified from the present study, it is one of *co-ordination and integration* between and among projects. The third section of Part I will go into more detail about this issue, but it is a theme worth raising here at the outset, and bearing in mind when considering the rest of this report.

Disclosure

The world of scholarly communications isn't a large one; many of the projects represented here are ones I've been following with interest for a very long time. As Director of the Canadian Institute for Studies in Publishing at Simon Fraser University, I am very well acquainted with the Public Knowledge Project (PKP); while I have no role with the project, members of the PKP's leadership team are my colleagues at SFU and indeed friends of mine. I hope my long history of being critical of Open Journal Systems (OJS) helps keep this report as objective as it can be. I have known Adam Hyde of the Coko Foundation for many years and have helped promote Coko and Adam's ideas generally. I am currently an advisory board member of the Rebus Foundation. Last, my financial sponsor in this project has been the MIT Press, who are a major stakeholder in PubPub.
– John W. Maxwell

The Landscape

Setting Context

What is Open Source Software?

In this report, "open-source software" (and "OSS") will serve as shorthand for the more inclusive term "free and open-source software." Defined very simply, we mean software that is developed in such a way that its source code is open and available online, and explicitly licensed as such. In their Guidebook, *It Takes a Village: Open Source Software Sustainability*, Arp & Forbes cite the Open Source Initiative's definition—"software that can be freely accessed, shared, used, changed, and/or modified"—and argue that this "fits well with the missions of organizations dedicated to documenting, preserving, and providing access to cultural and scientific heritage."[1]

Various claims have been made over the past two decades about the benefits of open-source software. Some claim that OSS provides a less expensive alternative to commercial software; some claim that the quality of the end product and/or the code itself is superior to proprietary software; some hold that OSS has a better chance of thriving over the long term because it can outlive the end of its institutional host or its commercial usefulness. Some hold that in areas like science and scholarship, OSS is part of an ethical imperative to keep academic work open and in free circulation. Most of these virtues are articulated as positives—but there is also a powerful negative incentive to promote and adopt OSS: the fear of lock-in and, ultimately, dependency on a corporate vendor.

In our research, we noticed that OSS is discussed and rationalized more often in terms of its *values* than its actual *practices*, so it behooves us to dig into how projects actually practice open-source. Brian Fitzgerald in 2006[2] wrote of a significant shift in how open-source software projects were being considered and operated. Fitzgerald noted that the rise of successful open-source software (which he called "OSS 1.0") was characterized by self-organized, Internet-based projects that gathered loose communities around sheer willingness to participate. Fitzgerald identified a newer mode, which he called "OSS 2.0," characterized by "purposeful design" and institution-sponsored "vertical domains," and much more likely to include paid developers. Fitzgerald's distinction is relevant to our study, as most (but not all) of the projects considered here fit within his "OSS 2.0" pattern.

1 Arp, Laurie Gemmill, and Megan Forbes. "It Takes a Village: Open Source Software Sustainability." LYRASIS, February 2018. p6. https://doi.org/10.7916/D89G70BS See also the Open Source Initiative https://opensource.org/faq

2 Fitzgerald, Brian. "The Transformation of Open Source Software." MIS Quarterly 30, no. 3 (2006): 587–598. https://doi.org/10.2307/25148740

Joel West & Siobhán O'Mahoney in a 2008 article, "The Role of Participation Architecture in Growing Sponsored Open Source Communities," made a further helpful distinction: between openness for the sake of *transparency* and openness as *accessibility.*[3] West & O'Mahoney saw that institutionally sponsored projects often tended to limit accessibility—which they characterized as community members' ability to make changes and participate fully in governance. Transparency, as a less radical virtue, meant that community members could merely see what design and development actions were being carried out.

All sponsors worked to achieve significant transparency in their open source communities, but sponsors varied considerably in the importance they placed on providing accessibility to external parties. This distinction provides a more nuanced understanding of the tension between openness and control.[4]

In our landscape analysis, we saw projects as ranged along such an axis of accessibility. At one end were projects being developed *transparently*, under an open-source license and which kept their code public in a Github repository; however, no significant contributions from outside the core team were encouraged. At the other end of the spectrum were projects that put community accessibility and participation first and for which a good deal of effort is made to encourage new contributors. Most projects fall somewhere between the two poles, but the tension between openness and control that West & O'Mahoney identify is an active one for many of the projects we discuss in this report.

The tension exists naturally enough because the current landscape is shaped by a blend of individual business goals with a growing ecosystem awareness that is concerned with the health of the overall sector, in a slow movement that is at least in part related to the rise of the Open-Access (OA) movement as an ecosystem-wide agenda. The idea that the publication and circulation of science and scholarship should not be controlled by profit-seeking corporations has led in recent years to a recognition that profit-seeking corporations, while possibly ceding ground on OA itself, had an almost total lock on the technological infrastructure that runs scholarly communication and publishing. Geoff Bilder, Jennifer Lin, and Cameron Neylon, in a widely cited statement, put it bluntly:

> Everything we have gained by opening content and data will be under threat if we allow the enclosure of scholarly infrastructures. We propose a set of principles by which Open Infrastructures to support the research community could be run and sustained. [5]

Elsevier's 2017 acquisition of bepress—an institutional repository system and company that was begun by faculty at the University of California, Berkeley—has proven to be a watershed moment in how many understand the scholarly communications ecosystem. Reporting on the bepress acquisition, Roger Schonfeld wrote:

3 West, Joel, and Siobhán O'Mahony. "The Role of Participation Architecture in Growing Sponsored Open Source Communities." *Industry and Innovation* 15, no. 2 (April 1, 2008): 145–68. https://doi.org/10.1080/13662710801970142

4 West & O'Mahoney, "Participation Architecture," 157

5 Bilder, G, Jennifer Lin, and Cameron Neylon. "» Principles for Open Scholarly Infrastructures." *Science in the Open: The Online Home of Cameron Neylon* (blog), February 2015. https://cameronneylon.net/blog/principles-for-open-scholarly-infrastructures/

> In a move entirely consistent with its strategy to pivot beyond content licensing to preprints, analytics, workflow, and decision-support, Elsevier is now a major if not the foremost single player in the institutional repository landscape.[6]

This moment gave an enormous boost to the idea of "community infrastructure." SPARC's executive director, Heather Joseph wrote that the event "sent a shockwave through the library community."[7] There is no doubt that the fear of enclosure—in this case of *infrastructure* rather than the content itself—is a key motivator today.

The fear of enclosure is certainly not the only force driving open-source development. Many funding agencies require that software developed under a grant be released as OSS in order to keep the fruits of their funding from disappearing into some corporation's vaults. There is also the hope, at least, of increased scale: a publisher or a library, interested to develop a bespoke tool, will find it difficult to justify the cost of development and maintenance if the only user will ever be itself. For many, the idea of open source implies a shared deployment model that distributes, if not the cost, at least the value, across a larger community.

OJS: Modeling publishing operations and open-source sustainability

With its conceptual origins in the late 1990s, followed by a first release in 2002, the Public Knowledge Project's *Open Journal Systems* (OJS) provides an early and lasting model for community-supported open-source infrastructure project. OJS was released as a downloadable LAMP-based[8] web application. It was adopted by a grassroots community of journal publishers and their (often institutional) supporters, one by one, until individual installations numbered in the thousands. Today, OJS is used by roughly ten thousand active journals[9] around the world and as such represents the most widely used piece of open-source publishing software.

Sustaining OJS over so many years has been—and remains—a challenge. The PKP has looked to support itself via blend of research and infrastructure grants, institutional subsidy, hosting and publishing-services revenues, and a large quantity of goodwill in its community. But OJS has survived (and even thrived) in an often dire-looking funding climate. It has survived because it addresses a very real need on the part of its large user base, one recognized not just by its users, but also by funding institutions, libraries, universities, and advocacy groups. OJS, having been originally conceived as a strategic *intervention* into the world of journal publishing, now shapes a significant portion of that world.

Less obviously, OJS has also succeeded in establishing a set of *de facto* standards for how a peer-review journal should be run. By modeling the workflows

6 Schonfeld, Roger C. "Elsevier Acquires Institutional Repository Provider bepress." *The Scholarly Kitchen* (blog), August 2, 2017. https://scholarlykitchen.sspnet.org/2017/08/02/elsevier-acquires-bepress/

7 Joseph, Heather. "Securing Community-Controlled Infrastructure: SPARC's Plan of Action." *College and Research Libraries News* 79, no. 8 (August 2018). https://doi.org/10.5860/crln.79.8.426

8 LAMP stands for "Linux, Apache, MySQL, PHP," the stack of open-source tools that defined the first major wave of web platform software Wordpress, Drupal, and OJS were all designed around this stack.

9 see Maron, Nancy. "Understanding the Audience of the Public Knowledge Project's Open Source Software." *BlueSky to BluePrint*, March 2018. https://pkp.sfu.ca/findings-from-community-consultation-2018/ See also, for detail: https://pkp.sfu.ca/ojs/ojs-usage/ojs-map/

and functions of a journal publisher in its software, OJS made explicit what was often implicit—or exchanged only in coterie groups. The result is that an entire generation of scholars has grown up with the OJS model. It now serves as a standard and a model for other projects, either as an exemplar to emulate, or as a point of departure for new design. As Chris Kelty pointed out to us, the pursuit of technical standards is also about standardizing practices; expert human labour is key to publishing.

Whether its longevity makes it the frontrunner in its class or ripe for replacement is a matter of opinion and perspective. We should not, however, underestimate OJS's contributions to how people *think about* publishing—and publishing software. In a very real sense, OJS defined the space that this report now seeks to analyze.

OJS as one project among many

Despite OJS's status as a kind of standard model against which other projects can be compared, there are many reasons why it makes less sense to do so. While the call today for community infrastructure may bring OJS to mind for many, there are also many other projects that define their scope, goals, and approach differently than OJS and, indeed, for the most part from one another. From a design point of view, OJS represents one possibility in a wide field of initiatives to create open-source publishing systems.

In such a large and varied landscape, developers and designers have carved out very distinct problem spaces and have different perspectives about which problems need to be solved and how exactly to go about solving them. Even among projects aiming to provide a full-stack journal-publishing platform, the aims and goals, and thus design decisions, vary widely enough that what we see is not so much competition over a particular niche as a proliferation of niches.

The result is a richly faceted landscape, but not one that lends itself to easy analysis. A proliferation of niches is both a boon and a curse. It is not, for instance, practical to "pick the winners" simply by looking at features and evident merits. It is not simple to connect the dots between manifest qualities like codebase, functionality, governance, and a project's ultimate chances for sustainability or success, because the open-source publishing landscape is a dynamic ecosystem, where the component parts—projects, funding sources, standards, and labour—exist in relation to one another and influence one another. The landscape needs to be considered as whole, and not just as the sum of its parts.

Defining scope

This report covers more than 50 projects, identified through a broad environmental scan conducted July–December 2018. Those 50-plus projects rarely, if ever, neatly line up for comparison. They all define their own scope, goals, and measures of success. This presents challenges for our analysis, and in many cases limits us to cataloguing these projects as opposed to evaluating them against an objective measure or standard.

We defined the scope of our environmental scan mostly negatively—that is, in terms of what we decided would be *out of scope* for this analysis. Our approach excludes the following categories:

- Closed-source. For the most part, all of the tools catalogued in this report have an accessible code repository (mostly on Github) and are released under an open-source license.
- Cloud-based services. We excluded online-only publishing services that do not offer their underlying code up for adoption. There are myriad such cloud services, especially in the burgeoning 'open science' community. But, as data-centric projects, the ability to download and adopt/adapt their code oneself is beside the point, and as such we decided these projects would not be part of our analysis.
- Research tools. There is a rapidly expanding genre of open-source software that supports workflow for researchers and labs. These tools are sometimes referred to as "research communications" tools, but we excluded these in the first place because we made a distinction between research communication and publishing as traditionally defined. Second, and more pragmatically, there are so many of these projects—any of which *might* be useful in a publishing context, but for the most part operating outside of such a context. We are reminded of Pluto, and the reasons why astronomers decided not to include it in the list of planets in our solar system.[10]
- Library infrastructure. We excluded digital library infrastructure such as Samvera, Islandora, and DuraSpace. These systems operate in an ecosystem of their own, and while they may in some cases underlie publishing software, their scope is outside this project.
- Baling-wire DIY projects. There are innumerable 'publishing solutions' that involve gluing together one or more open-source tools with a handful of automation scripts (often using a conversion tool like Pandoc and leveraging Github as a content management system). While we applaud these efforts (and have built them ourselves!), such ad-hoc toolchains do not in themselves constitute OSS projects on the scale with which we are concerned here.
- Dormant. We initially gathered but later culled a number of projects

10 In his popular book, *How I Killed Pluto and Why It Had It Coming* (2010; Random House) astronomer Mike Brown tells the story of Pluto's 'demotion' from the status of a planet to a 'minor planet.' The discovery of Eris, and indeed, of thousands of such objects orbiting the sun, implied that either the number of planets in our solar system would grow astronomically (pun intended) or we would need a new, tighter definition of "planet." In 2006, astronomers chose the latter course.

that have had active lives and communities but did not appear to be active in the past two or three years. While the code for these projects is still accessible, the lack of an active developer or a sustaining community suggests that supporters have moved on to other projects. In some cases, a project will have been explicitly superseded by another. In other cases, developers will have left a project behind to join another. The latter phenomenon hints at a possible lesson: the mere existence of open-source code without an active developer who cares about it is not worth much, in practice.

Mapping the Landscape

Some axes of analysis:

Across the 52 projects that we have catalogued here, there is great variation across a number of different axes. The following subsections provide some possibilities for subdividing the landscape along some of the more obvious lines.

Journal publishing & book publishing

Some projects we catalogued are straightforwardly oriented to journal publishing. Some (especially given the Mellon Foundation's recent funding moves[11]) are oriented to monographs and books. But a substantial number occupy a space in between—agnostic with regard to journals or books, and sometimes reaching for new forms altogether.

Centralized vs distributed models

Several projects we catalogued are designed around a central hosting model where there is considerable value in how the project host supports the software centrally; a prime example is Fulcrum, developed and hosted by the University of Michigan Library and Press. Other projects, like OJS, are designed around a distributed model where anyone can download and deploy the software. An increasing number of projects seem to anticipate a hybrid position in which any number of third-party hosting/integration partners will take care of deployment (and effectively become partners in the operational life of the software). None of these projects, by virtue of their open-source licenses, are strictly constrained to one or other deployment model; our observations here are about how the development is unfolding currently.

Old projects and new projects

As might be expected, we were able to catalogue a wealth of new projects, and

11 Waters, Donald. "The Monograph Is Dead! Long Live the Monograph!" presented at the Jisc-CNI Leadership Conference, July 2, 2018. https://www.slideshare.net/JISC/the-monograph-is-dead-long-live-the-monograph; Maxwell, John W., Alessandra Bordini, and Katie Shamash. "Reassembling Scholarly Communications: An Evaluation of the Andrew W. Mellon Foundation's Monograph Initiative (Final Report, May 2016)." *Journal of Electronic Publishing* 20, no. 1 (2017). http://dx.doi.org/10.3998/3336451.0020.10

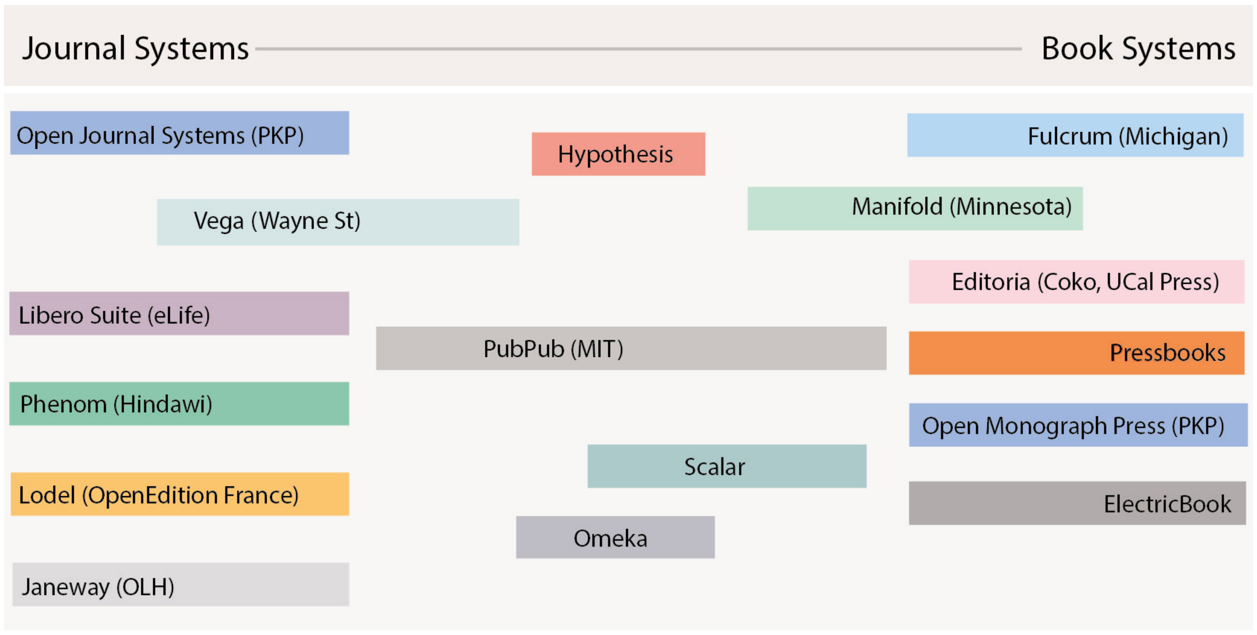

Figure 1: Examples of journal vs book orientation (selected projects)]

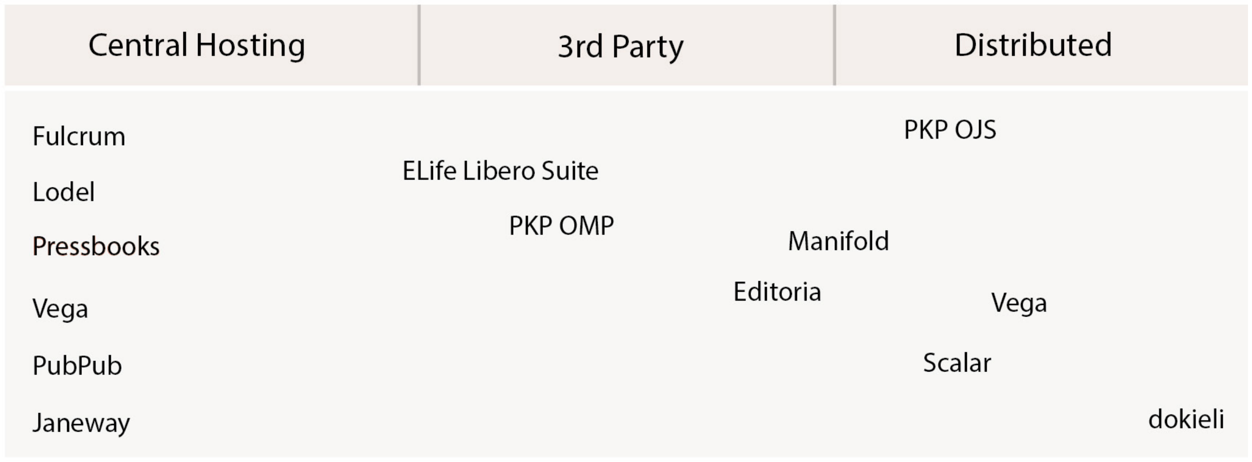

Figure 2: Centralized vs distributed deployment (selected projects)

a smaller number of older, more established projects. OJS is the longest-running project we catalogued, established in 2002. Some other notable projects are the bibliography manager Zotero (est. 2006), the French journal platform Lodel (est. 2006), the conversion tool Pandoc (est. 2007), the authoring tool Omeka (est. 2008), the annotation platform Hypothes.is (est. 2011), and the Math typesetting system MathJax (est. 2011). By contrast, *fully half* of our catalogue has emerged since 2015, with more than a dozen of these projects having their first release since 2018.

This, again, makes comparison difficult. Brand new, bursting-with-promise projects simply aren't directly comparable to those that have weathered time, competition, and the ongoing demands of users. Conversely, longevity tells a story of fitness, but is difficult to make generalizations. A 'graveyard' of old and abandoned projects does not really exist, as Github only emerged as a common platform for software development projects in around 2012–2013. Projects older than that, even if the source code is still available, are not easily findable.

Functional scope

Very few of the projects we catalogued even do the same things. Some are attempts to create end-to-end functionality for an entire publishing process; an example is the Libero suite from eLife. Others offer very specific functionality, but may be usable in concert with other components; the best example here is Hypothes.is, which does one thing—annotation—very well and can be integrated in a variety of contexts.

To help visualize the functional scope of various development agendas, we propose a hypothetical publishing workflow that covers a number of stages in order to show how various projects address different functional areas. But we must emphasize one serious caveat: even though different projects may address the same workflow stages in this diagram, they most likely do so differently, with different boundaries and different goals. Our focus here is with software development priorities, rather than "features" *per se*. We thus offer the following diagram for illustrative—*but not comparative*—purposes:

Operational details

The projects we catalogued also differ in development features, languages and frameworks, and licenses. Some are well supported by external funding, some struggle to maintain financial support, some (including some important projects) are effectively unfunded. We offer the following summary data, again for illustrative purposes:

Development language: Thirty of out of fifty-two projects were written using *JavaScript*. Nine are in PHP, seven in Python, and the rest in a variety of languages including Ruby, Haskell, Go, and XSLT.

License: Seventeen projects are released under the MIT License; seven under the GPL v3, seven under the GPLv2, and seven using a BSD license. The remainder use AGPL, Apache, or ECL licenses. Comparing these numbers with a 2018 report by Ayala Goldstein 2018, the proportions here are close to the proportions for Github as a whole.[12]

Funding: About a dozen of the projects we catalogued claim *multiple funding agencies*; this unsurprisingly tends to correlate with the age of the project. Another dozen projects appear to have *no funding at all*—apart from the developers' time on the project. At least fourteen of the projects have received funding from the Andrew W Mellon Foundation.

12 Goldstein, Ayala. "Top 10 Open Source Licenses in 2018: Trends and Predictions." *Whitesource*, December 13, 2018. https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions

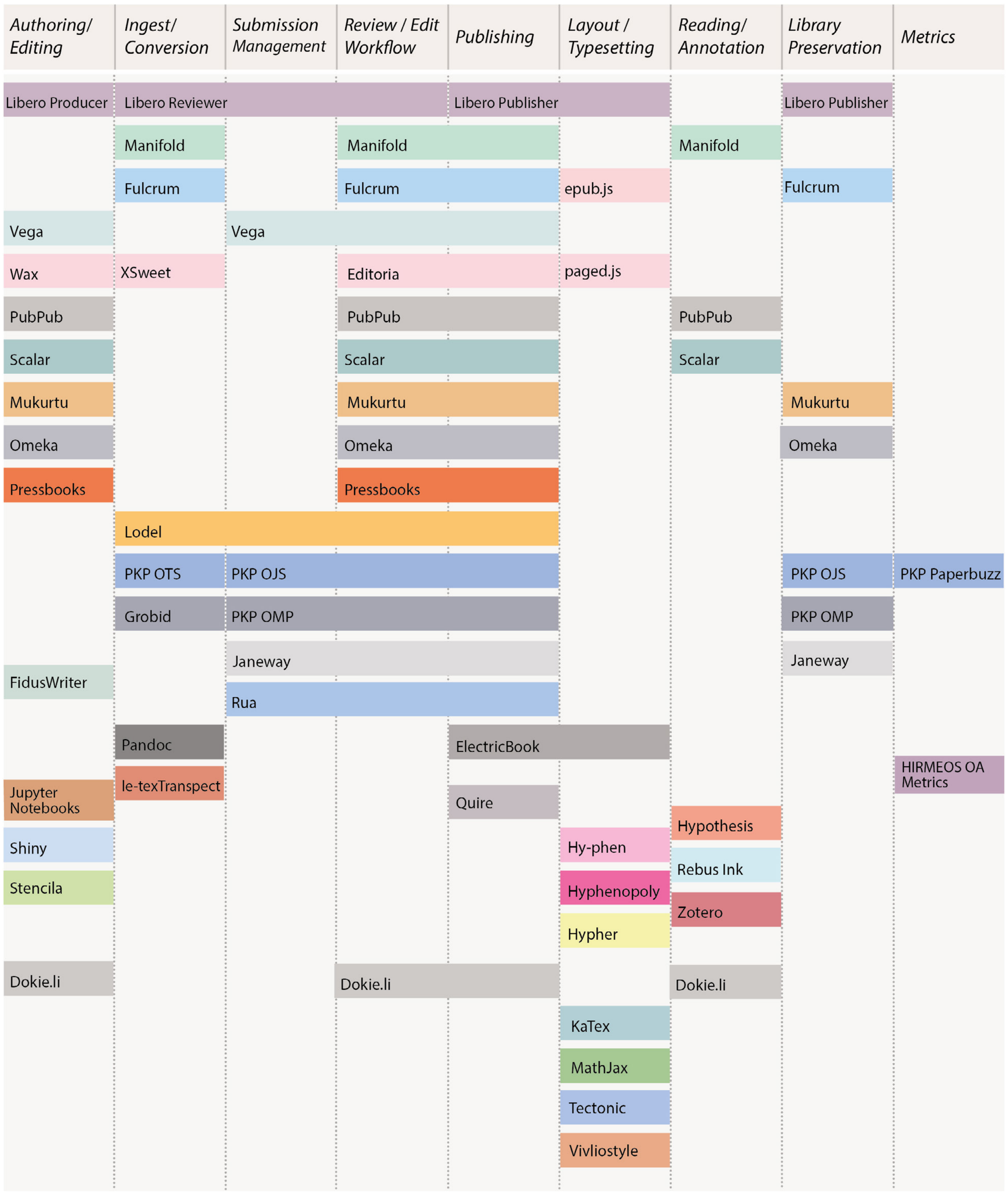

Figure 3: Software development across hypothetical workflow stages

Traditional functions, new capacities

It would be easier to examine publishing software if we all weren't simultaneously in the midst of reinventing publishing itself. If publishing functions and the scope definition of journal or monograph publishing were stable over time, it would be more straightforward to judge software offerings against a functional standard. But, at the same time that we are re-building publishing infrastructure (in both open-source and proprietary contexts), we are broadly at work redefining publishing itself, as well as the forms and genres that define scholarly communications.

Journal publishing, while the most transformed by a three-decade shift to online distribution, at least sees some stability in its essential forms. A great deal of the innovation in journal publishing is concerned with the drive to scale and production efficiency, leaving the basic form of the *article* alone (there are of course exceptions, as in eLife's Reproducible Document Stack and similar data-rich, interactive formats).[13]

Book publishing is another story, where a key source of innovation comes from the desire to produce and publish *interactive scholarly works* that have comparable size and significance to a traditional book, but share little with them production-wise. The latter shift has been identified and encouraged by the Mellon Foundation in recent years.[14]

At the same time, the affordances of web publishing have spawned a host of publication formats and platforms that are *web native*—neither journal nor book—that proceed less from a sense of traditional forms than a sense of what can be done, quickly and elegantly, online.

These trends complicate our landscape analysis. Some of the projects we catalogued seek very straightforwardly to model existing publishing practices while extending their efficiency or flexibility through digital media. OJS is perhaps the original case, aiming to pave the way to a fluid, open-access ecosystem. Its original design principles sought to embody existing best practices in journal publishing. OJS was not designed to be disruptive; rather its goal was to allow journal publishers to move their existing operations into an online, indexed environment.

An example of modeling existing publishing practices in book production is Editoria, developed by the Coko Foundation, the University of California Press, and a community of other interested academic publishers. Editoria is an editorial and production system for scholarly monographs designed to provide a web-based, collaborative platform with much more output flexibility than traditional proprietary tools offer. Editoria's aspiration to be a drop-in replacement for existing tools makes it an ambitious development effort, but perhaps a necessary one if uptake in traditional university press operations is the goal.

While tools like OJS and Editoria serve established publication models,

13 See: Jupyter Notebooks; Shiny; Stencila.

14 Waters, "Long Live the Monograph!"

many of the tools we catalogued seek to break new ground and open up new possibilities in scholarly communication. MIT's PubPub provides a full-featured platform for research teams to communicate with colleagues and the wider world; PubPub could be used to publish traditional scholarly works (journals, books), but it opens up a faster, more reader-centric modality that isn't neatly contained by current publication norms. The University of Minnesota Press and CUNY's Manifold Scholarship can hold scholarly monographs within it, but the point of the tool is to facilitate and capture the ongoing discourse around a book, rather than just the book's content. Well-established tools like Omeka and Scalar exist to break new ground with the integration of multiple media and non-linear content organization.

Special-purpose components—from web-based word processors (Wax, Texture, and FidusWriter) to typographic toolkits (Hyphenopoly, KaTeX) and annotation and reference systems (Hypothes.is, Zotero)—often are agnostic to the publishing formats or genres they can serve, with the exception of assuming the Web as a common platform. It is worth noting that we also include contemporary examples of print production tools (Paged.js, Vivliostyle).

Technological approaches and trends

The software projects surveyed here represent a variety of approaches to contemporary problems, and as such provide a rich snapshot of contemporary thinking about publishing and software strategies. While the vast majority of these projects are *web-based* in one way or another, they vary greatly in their priorities and the bids they make to exist in a much larger ecosystem. The following are some significant trends we noted:

Approaches to XML

Most of the software we surveyed involves representation of text: for authoring and editing purposes and for display and publication. XML is central, in one way or another, to almost all of the projects. But what does that mean, exactly? Two dominant approaches to XML are evident: the first, employing the JATS XML schema for rich semantic markup and robust in-document metadata, seems to be a popular choice with projects focused on journal publishing workflows. The Texture editor from the Substance Consortium (including eLife and PKP) provides an excellent open-source, JATS-based authoring and editing platform which can then be incorporated into other tools. ELife's Libero Producer is designed around Texture, building a JATS-native[15] editing interface right into the core of eLife's platform. OJS, which for most of its history has eschewed dealing with the text directly (opting to move .doc and .pdf files through its review workflow), now allows Texture integration as an option, and PKP seems enthusiastic about Texture's development and future. Janeway, designed for the Open

15 Texture's XML file format is .dar, which is an encapsulated collection of XML content (compliant with a Texture-specific JATS subset) and its related assets, plus a manifest file listing the contents. See https://github.com/substance/dar

Library of the Humanities platform, is also based on JATS and seems poised to adopt Texture as well. This is a potentially important moment for JATS XML. While a 'standard,' JATS has not enjoyed actual standardized practice, because JATS-based workflows are typically buried in proprietary toolchains owned by corporate publishers. The emergence of an open, common editing tool for JATS is a welcome development for XML-based publishing ambitions.

The second major current of XML development is the use of web-native HTML as the basis for content and workflow. Owing to the ubiquity of this format and the wealth of readily available tools and standard ways of working, many of the projects we surveyed have opted for an HTML-first approach. This is true of journal-friendly projects like PubPub and Vega, but is especially the case with the more book-oriented projects such as Fulcrum, Manifold, Editoria, Pressbooks, and Scalar. In an HTML-based workflow, rendering in the browser comes more or less for free, and the associated EPUB standard (which includes HTML as its core text representation) provides a handy distribution or import/export format. More interestingly, authoring and editing tools for HTML are by now in their third or fourth generation, and sophisticated software is not hard to come by. An emerging open-source toolkit, ProseMirror has already seen significant uptake on the web (major news sites like New YorkTimes and The Guardian have reportedly built editorial tools around ProseMirror) owing to features like collaborative editing. ProseMirros is found in PubPub, Coko's Wax editor (part of Editoria), and the science-oriented FidusWriter. There seems to be increasing interest in ProseMirror as an adaptable foundation for building specialized HTML editing environments.[16]

A third alternative, which puts *markdown* before markup, is seen in some production systems such as ElectricBook and Getty's Quire. The markdown approach relies on a simplest-possible authoring environment (in a text editor) and up-converting to HTML or other XML formats. Markdown is also a straightforward import format for tools like Manifold, PubPub, and Pandoc. ProseMirror seems able to work as easily with markdown as with HTML, so the apparent distinctiveness of a markdown-based workflow may fade over time.

A big part of the appeal of browser-based XML is the ability to generate final typeset output there. For reading on the web, this is obviously the case, but it is easier than ever before to produce paginated output from the browser as well, and to shed the necessity for an additional PDF-rendering tool in one's workflow. Two major projects take browser-based pagination and layout seriously by developing feature-rich JavaScript rendering engines for page-based layout. Vivliostyle is the older of the two and has been used in a number of publication projects over the past three years. A newer project, Paged.js, seems to be picking up a large community of interest. We include here too a handful of smaller, more specific-purpose JavaScript publishing tools, including three competing JavaScript libraries for doing proper hyphenation and justification

16 See, for commentary, Triglav, Jure. "Open Source Collaborative Text Editors." *A Case for Spaceships* (blog), May 7, 2019. https://juretriglav.si/open-source-collaborative-text-editors/

(H&J) in a browser-based environment—one of the last hurdles to making online typography (and thus reading experiences) rival that of print. The world of browser-based pagination and layout should become even richer as parts of these custom toolkits become better support by native browser features, thus relying less on custom JavaScript code.

LaTeX deserves a mention here. One of the original OSS publishing tools (LaTeX, and TeX especially, predate the term "open-source" by many years), LaTeX is still alive and well in scientific publishing. Its support for equations and formulae remains hard to beat, despite efforts to move LaTeX's features into more modern environments. In our survey, LaTeX appears in only a few cases. We examined here one contemporary platform, Tectonic, which seems to be an easily adoptable typesetting tool. We considered including Overleaf, the leading commercial LaTeX-based production system, as their codebase is open-source and accessible on Github, but we ultimately decided to remove it as it seems to have no substantial interest beyond Overleaf's own application. LaTeX also appears in a few web-typography tools aimed at math typesetting: KaTeX from the Khan Academy, and MathJax, both of which aim to provide a browser-native math typesetting system that does what LaTeX does, and indeed can speak LaTeX.

Conversion and ingestion strategies

Despite the maturing contexts of XML in publishing, it appears to be a largely unchallenged fact that "authors will write in Word." Word processor documents, despite the advent of XML file formats over the past decade, are just not structured documents, because the scope of possibilities that an author can express in a tool like Word is not constrained by any schema. Further, the vast legacy of online publishing has been the proliferation of PDF files—again, not a structured content format. So any publishing system that attempts to leverage structured content while allowing content to come from unstructured sources must have a strategy for ingesting these source documents and making sense of them.

This problem is as old as XML—indeed as old as SGML—and toolchains to solve the problem as numerous as the grasses; it appears that people continue to build these today. The emergence of XML-based word-processor file formats at least has made parsing a bit more straightforward, allowing XSLT to be used to at least take the original document apart. In our landscape survey, we have catalogued at least half a dozen projects dedicated to import and conversion, and at least as many larger projects have ingest tools built into them.

The traditional way to convert legacy documents is to parse them—either via XSLT or some other way of reading the native file format, and then attempting to make reasonable guesses about what the formatting means: the big, boldfaced line at the beginning of an article is likely the title, for instance. If the original document was formatted using named paragraph- or character styles, so much the better. Some of these parsing tools are mature and can handle a good many variations. Pandoc, for instance, is a robust conversion utility that has been in development for over a decade, with support for dozens of input and export formats. It is usable as a tool on its own, but it is also incorporated as a library or a component in several of the tools in our survey.

A traditional strategy for managing conversion from legacy formats is to constrain the scope of possibilities. Building a conversion tool around documents that consistently look like journal articles is easier than building a general-purpose converter. PKP's Open Typesetting Stack[17] has been designed using this approach, as is OpenEdition's Lodel. Open Typesetting Stack is composed of a series of tools that are designed to take apart journal articles: front matter, body text, bibliographic references, and so on.

A newer approach altogether is to forgo parsing the internals of a file and instead pay attention to the visual and presentational characteristics of a PDF. Grobid, a machine-learning tool trained on a corpus of many thousands of journal articles, exemplifies this strategy. The latest versions of PKP's Open Typesetting Stack include Grobid in its arsenal. Machine-learning tools improve over time and over larger datasets, so it seems likely that this approach will become common, if not dominant, in large-scale conversion and ingest of journal articles. Grobid—like several other tools (including le-tex Transpect, Lodel)—uses the Text Encoding Initiative's (TEI) extremely rich and flexible descriptive XML tagset as an intermediate conversion target before normalizing to JATS XML for publication purposes.

Workflow modeling and management

Scholarly publishing is typically characterized by formal editorial review processes, including blind peer review. Modeling and capturing these formal review stages in software is a hallmark of scholarly publishing applications. OJS first established a formal model for peer review workflow nearly twenty years ago, designed around a hierarchy of editorial authority, explicit hand-offs from stage to stage, and a series of automated email reminders keeping every member of the process on task. OJS's fine-grained, formal peer review has clearly stood the test of time (the model was made more modular in OJS 3), but developers and aspirants have been re-thinking and re-building editorial and review workflows ever since. The most recent generation of publishing software carries on this tradition, and re-designing workflow management is a feature in most of the projects we examined.

Some approaches aim to make submission and review simpler. PubPub, for instance, aims to make collaborative reviews easy and intuitive. Vega takes a similar approach, establishing a new conceptual vocabulary around the review model. Manifold brings robust commenting and annotation to its review process,

17 PKP's Open Typesetting Stack is based in part on Martin Eve's now-dormant meTypeset conversion tool. Both are unfortunately misnamed, as they aren't typesetting tools at all.

perhaps more in the spirit of ‘open review.’ Ubiquity Press, while relying on OJS as the core of their journal-publishing platform, have made customizations for article review and have built an entirely different system, Rua, for managing book editorial processes.

The Coko Foundation and its partners have taken a somewhat different approach by building a layered and modular framework for workflows. Coko’s PubSweet framework exposes a set of components for integration. Specific applications—like eLife’s Libero Reviewer or Hindawi’s Phenom—configure these to the specific business/editorial needs of their publishers. EuropePMC and Wormbase’s micropublications framework also manage submissions this way. On the book-publishing side, Editoria is also built on top of the PubSweet framework, as is the BookSprints platform. As such there are at least six different workflow applications based on the PubSweet workflow system, and Coko’s promise is that many more are possible.

Whatever the specifics of workflow management in various contexts, it would appear that many people still see this as a problem that needs a solution—or indeed more solutions. It may be the case that workflow modeling is something that resists being *solved* once and for all. In an interview, one of the PKP team quipped that once some of the newer projects have been around for as long as OJS has—and if they are to serve a diverse user base—their simple workflows will need to evolve to serve those diverse needs. The many attempts to address workflow models in the current catalogue seems to support this view.

Innovating new possibilities

Many of the projects in this survey also seek to push the envelope, to expand the possibilities of digital scholarly publishing. These range from infrastructural innovation to blue-sky revolutionary thinking—like dokie.li’s decentralized, distributed authoring/publishing project, which is part of a rethink of the entire World-Wide Web from a linked-data perspective. Most projects we surveyed are a little more conventional, but many break new ground in thinking about how scholarly communications actually happens.

The University of Michigan’s Fulcrum project, for instance, makes a significant structural change in how we think about infrastructure. Fulcrum does not take great strides with user interface, but by building a robust, media-friendly ebook platform on top of the Samvera repository, developing robust metadata linkages between books and media objects, and integrating a set of modular tools for displaying and embedding these, Fulcrum has potentially emerged as a major new platform for digital book distribution, one that several other publishers seem to find attractive. Fulcrum potentially changes the ecosystem for scholarly ebooks, making media rich content workable and discoverable, at scale.

The University of Minnesota & CUNY Graduate Centre’s Manifold Scholarship also elegantly integrates a set of good ideas, while pushing out the post-production scope for book-length works. Manifold aspires to gather the discourse around a book—review, commentary, annotation, and even social media discourse—and collect it within the book itself. The result is that books expand over time as they gather their surrounding discourse. Manifold was initially designed as a monograph publication tool but has already found applications in open educational resources and in critical digital editions, owing to its reader-focused feature set.

MIT's Knowledge Futures Group offers PubPub, a scholarly publishing tool that hosts journals, books, reports, and related content types, but seems poised to gain a devoted audience by making it incredibly easy for a research lab or team of like-minded scholars to collaboratively develop and publish media-rich content on an ongoing basis. It is early yet to tell if PubPub will evolve into a research-publishing platform or a turn-key publishing alternative. Vega, designed by Cheryl Ball after many years of publishing the *Kairos* journal, aims to bring multi-media authoring and collaboration into the centre of scholarly discourse. Vega has been long anticipated by those inspired by the promise of its model; it appeared in alpha release in early 2019.

Omeka has been in development for more than a decade already, but it, as well as ANVC's Scalar, and Washington State University's Mukurtu pushes on the boundaries of what a book might be in a natively digital mode. Omeka, Scalar, and Mukurtu have all been focused on scholars and researchers first, as opposed to presses, but the wealth of content and projects published on these systems already (including the *Ravenspace* project from the University of British Columbia and University of Washington Presses, which draws in ways on all three) means that these platforms are part of the discourse around the nature of the book in an online context. Stanford University Press's embrace of Scalar-based projects is evidence that this platform is being taken seriously by traditional publishers.

An emerging genre of writing tools—exemplified by Jupyter Notebooks, RStudio's Shiny, and the Stencila project (part of eLife's Reproducible Document Stack initiative)—integrates written documentation with live code and data in a publishable interactive environment. A researcher can write an article, incorporate a dataset, and feature live code snippets and data visualizations in the body of the article. Shared or published online, a reader can then interact with the data or the code directly, effectively bringing into play a richer way of constructing and communicating a scholarly or scientific argument. Shared between two researchers, these tools are clever enough; all three projects are pushing towards much broader scale publication of interactive documents.

Two well-established projects—the Hypothes.is annotation system and the Zotero reference management software—plus one newer one, the Rebus Foundation's Ink platform for research-based reading—deserve mention here too. These are not publishing tools *per se*, but they serve critical parts of the publishing and scholarly ecosystem. Hypothes.is, while not being the only approach to annotation represented here, has established a standard approach to web annotation that now appears to be essential. Zotero, which as a networked platform is much more than the personal reference manager most people use it for, is the primary open-source platform for large-scale bibliography handling. Both Hypothes.is and Zotero should, at this point in time, be judged in terms of their integration with other applications in the publishing and scholarship ecosystem; certainly no one should be developing in this space without considering the contributions already made by these tools. Which brings us to the Rebus Foundation's Ink project: funded by a grant from the Mellon Foundation, Ink is an experiment in developing a better integrated environment for scholarly reading, reference and document management, and annotation. Ink's development is made with tools like Hypothes.is and Zotero already established; if it comes to fruition, it should shift the thinking around what happens to scholarly publications when they reach readers, an aspect somewhat under-developed currently.

Prospects

Beyond the individual projects in our catalogue and the individual contributions they make, we also have to consider the larger ecosystem: how these projects relate to one another (both formally and informally), how they might be sustained over time, and how the higher-level goals of furthering scholarly communications are actually addressed by individual efforts and approaches.

Two larger-scale themes seem apparent to us after looking at the details for many months. The first has to do with the problem of siloed development. Many projects we surveyed operate largely in isolation from one another. The goals of collaboration, interoperability, and integration are very secondary to the specific, internal goals of each project. Incentives for collaboration between projects are few, even though there is a general recognition that where possible, collaboration, standardization, and even common code layers can provide considerable benefit to project ambitions, functionality, and sustainability.

The second theme has to do with the organization of the community-owned ecosystem itself: what are the forces—and organizations—that serve the larger community, that mediate between individual projects, between projects and use cases, and between projects and resources. The enormous plurality of approaches and strategies is both a positive (in the sense that the scholarly project more generally treats pluralism as a good), and a negative (plurality tends to work against the scale that is needed for efficiency—and indeed sustainability in a market paradigm). Neither a chaotic plurality of disparate projects nor an efficiency-driven, enforced standard is itself desirable, but mediating between these two will require broad agreement about high-level goals, governance, and funding priorities—and perhaps some agency for integration/mediation.

Collaboration and its benefits

Funding bodies—and especially substantial government and foundation grants—have been used substantially to support the development of many of the projects in this survey. Most such funding, though, is derived from a research-funding model that prioritizes new knowledge creation. It rewards the novel, the exceptional, and the singular. There is, by contrast, relatively little available funding for long-term development, and little funding, or incentive, for collaboration across initiatives. The result is that individual projects end up competing for the same funding sources, potentially at cross-purposes, and at the risk of unsustainability.

A culture of competitiveness and prestige in funding—itself inherent in academic research funding structures—privileges innovation over stability for many projects. From a funder's perspective, the return on investment (ROI) is more obvious where innovation is the goal than in long-term infrastructure investments. From a awardee's perspective, the flip side of this is prestige. In *Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure*, Nadia Eghbal noted:

> Older projects have a harder time finding contributors, because many developers prefer to work on new and exciting projects. This phenomenon has been referred to as "magpie developer" syndrome, where developers are attracted to "new and shiny" things.[18]

Long-term survival, though, is not shiny. It's just hard. In a pure market-driven environment, sheer perseverance, pluck, and luck are what lead to sustainability. But if we are talking about community-supported infrastructure, what are the equivalent dynamics? What would a serious funding environment look like without competition for resources at its heart? What would project funding look like if it prioritized community governance, collaboration, and integration across a wider ecosystem?

But aren't open-source projects collaborative by their very nature? If the code is available to all, then anyone who wants to contribute or integrate a project is free to do so. This framing, however, underplays the role of labour and active attention. For instance, OSS projects eventually end when they run out of steam—enthusiasm and energy on the part of developers and supporters, or else get swept aside by newer or better resourced projects that attract developer time and supporter attention. The *transparency* part of the OSS rationale suggests that, because the code remains available, in theory there are no dead projects,

18 Eghbal, Nadia. "Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure." Ford Foundation, July 14, 2016. 42. https://www.fordfoundation.org/about/library/reports-and-studies/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure

only dormant ones that could still be forked or reanimated, perhaps by a group of interested users. But this underestimates the scale and cost of OSS 2.0 projects. Indeed, what truly keeps OSS projects alive is communities of people who *care*—either as developers, supporters, or as users. Such care is not a cheap commodity; the OSS landscape is no Field of Dreams.

Partnerships and collaborations, whether among peers or among groups with aligned interests, are important to keeping more energy—and thus resources—flowing. The vast majority of projects we studied are small; their development communities are often fewer than a dozen people, and the direct interest in a project rarely extends beyond the institution that sponsors it. If the care and up-keep of projects could be extended to multiple groups, multiple institutions, then not only is there a larger and more diverse set of people who care, but opportunities for resourcing increase, and also, when one group's priorities inevitably shift, it is less likely that a project is simply abandoned.

The Coko Foundation's strategy is based on this idea. Coko—whose founders Adam Hyde and Kristen Ratan both had a wealth of experience with trying to sustain projects in the past—set out to build a community of interested institutions first, and then to design a set of software components that could work across use cases. Coko's initial set of institutional parters—including eLife Sciences, Hindawi, the University of California Press, California Digital Library, EuropePMC, and the Wormbase project—represent a very diverse set of needs and use cases. But Coko's foundational framework, PubSweet, is at work in all of these contexts.

Representatives from eLife and Hindawi spoke, at the Society of Scholarly Publishing conference in San Diego in May 2019, about how they had built very different review and editorial workflow software—with different business needs and user scenarios—on top of the common Coko codebase. This kind of collaboration both strengthens the core community and also provides more support to individual participants; for each new participant that joins the community, there is less work to be done on foundational pieces, leaving more time and resources for integration and customization.

A challenge here is in designing software for a broad, non-specific application (that can be built upon by others). Who will fund such an initiative in the first place, and who will direct the design? Coko has apparently succeeded with its PubSweet components, but at the cost of considerable community-building effort. By contrast, OJS 3 was developed with modular workflow system so that a partner developer could customize the way it works. But without PKP spending (that is, prioritizing resources) substantial time actively promoting this facility to potential partners, the software's capacity for more specialized configuration goes mostly unused. Another example is the Texture editor, which has the potential to become a standardized JATS editing and typesetting environment, yet its 'consortium' consists of just two organizations. Who will direct its design going forward? Can Texture realistically become a general-purpose JATS editor under such circumstances?

A slightly different story surrounds the ProseMirror editor framework, which isn't developed by a consortium. Rather, ProseMirror is the work of just one developer, Marijn Haverbeke. ProseMirror's Github repository has twenty-odd contributors, but Haverbeke also runs a crowdfunding campaign that has more than 400 contributors—among them many scholarly communication organizations. ProseMirror has found its way into many other projects, including PubPub and Coko's Wax editor. Notably, ProseMirror itself isn't the end product for this community; ProseMirror is the framework upon which others build their products.

Collaboration in these projects does not just mean alignment around a single tool; it often means approaching development as a stack of software layers that work together, some of which might be one's own primary concern, and others drawn from the community. But while it may make conventional sense to a downstream developer to re-use an existing modular component, who is responsible for doing the upstream work? Or for working on the generalized design and specification work for it?

It seems to us that there is an opportunity, either via funding mechanisms or by some agency for community stewardship, to provide clearer incentives for collaborative development, rather than projects proceeding from singular vision to an isolated codebase. If the goal of community-owned infrastructure is to succeed, then structural attention needs to be paid to the integration of projects, goals, and development efforts across the ecosystem. Nadia Eghbal noted that, "Not unlike technology startups, new digital infrastructure projects rely upon network effects for adoption."[19] The example of big publishers like Elsevier and technology companies like Digital Science shows that such network effects, and the integration of components across myriad workflow touchpoints, is key to succeeding in an interconnected world.[20] In an interview, anthropologist Chris Kelty pointed out that since 'infrastructure' layers can be harder to fund than 'applications,' Elsevier's focus on integration provides a major advantage.

Ecosystem integration and role(s) of service providers.

The integration of various functional components needs to be seen not just from the perspective of development, but also deployment. Connecting usable software with publishers and users is not straightforward, and there are—again—a variety of approaches within the group of projects we've examined.

The PKP's OJS has always embraced a DIY, download-and deploy methodology, and this has been key to a great deal of this platform's adoption. OJS' success in promoting Open Access (OA) publishing is partly because anyone who wanted to start an OA journal could do so, simply by installing OJS, setting up an editorial board, and publishing. Relatedly, OJS's significant popularity in the Global South is partly due to the self-contained nature of the software; any institution capable of running a webserver became able to participate in the scholarly communications environment. This model is very much in the spirit of first-wave open-source software; indeed, OJS's deployment model has much in common with WordPress.

At the other end of the spectrum, centralized commercial publishing and hosting platforms serve a different kind of end-user. Ubiquity Press, using the same software platform, built out a different service model around OJS, appealing to a different set of user needs. Both the Fulcrum project from Michigan and MIT's PubPub are hosted centrally, where the open-source software platform relies on a set of services that can only be effectively delivered with the support of the host institution: preservation strategy, identifier and discovery layers, and so on.

We have of course seen myriad examples in the middle range of this spectrum and in hybrid approaches. PKP, for instance, has put considerable energy into nurturing (and educating) libraries to become local hosting and deployment services for OJS. As well, PKP Publishing Services now offers fee-based hosting and integration. A hybrid approach to deployment has served Hypothes.is as well. The download-and-go model has allowed thousands of individual users to integrate Hypothes.is with their scholarly practice, while the organization has actively pursued publishers and platforms to integrate the annotation service natively. Across the landscape we've surveyed are a host of perspectives on this issue, and the challenging questions it poses: *If we rely on publishers to download and host themselves, will we scale the community to meaningful levels?* And, conversely: *If we offer centralized hosting, does that put us in market competition with organizations that would otherwise be our peers and partners?*

A recurring theme in conversations with several projects has been the expectation that a layer of third-party *service providers* would emerge in the coming years, allowing the challenges of deployment to be mediated by commercial (or non-profit) partners who would provide hosting, customization, and integration for a service fee. Such partners would become, in effect, development partners in the software, and help expand the community of stakeholders around a project.

This sounds encouraging, but who exactly will these third parties be? One answer might be libraries and university IT service departments, as in PKP's model. Another possibility is that commercial web-hosting providers could specialize into this market, offering scholarly publishing tools in addition to the usual WordPress or Drupal content management systems. A third possibility is a class of purpose-built providers who emerge around specific publishing communities, as Ubiquity Press did. Indeed, the rhetoric of community-owned infrastructure leads to a vision of a network of integration partners who make

19 Eghbal, "Roads and Bridges," 45.

20 Schonfeld, Roger C. "Strategy & Integration Among Workflow Providers." *The Scholarly Kitchen* (blog), November 7, 2017. https://scholarlykitchen.sspnet.org/2017/11/07/strategy-integration-workflow-providers/

all these tools work like a unified network, rather than as a lot of competing projects.

As much as we like this idea, and we can imagine what this might look like once established, it is far from clear, in the summer of 2019, how we get there from here. John Chodacki and colleagues, in their guidebook, *Supporting Research Communications*, paint a picture of a fragmented and somewhat confused research communications ecosystem, with as many differences as commonalities, even amongst supporters.[21] In such an environment, the development of a coordination and integration layer across diverse publishers and diverse functions will take effort, money, and initiative. It isn't something that will magically emerge from the current landscape.

Encouragingly, people are talking about this. The *Joint Roadmap for Open Science Tools* (JROST) initiative in 2018 launched with the observation that "we are aware there are obvious synergies that are not being pursued, and likely many others still waiting to be discovered" and talked of common goals, consolidation of effort, shared governance models, and standardization.[22] In summer 2018, Code for Science & Society's *Open Source Alliance for Open Scholarship* (OSAOS) working group released a report of their discussions, especially outlined a possible vision for how funding could be better coordinated to support open infrastructure.[23]

In the spring of 2019, the launch of the *Invest in Open Infrastructure* (IOI) went further, adopting a more action-oriented agenda that pulls together research on scholarly infrastructure broadly, with a focus on collaboration and interoperability, and seeking solutions for funding to sustain it.[24] One of the first concrete outcomes has been Educopia's report on the 2019 *Mapping the Scholarly Communication Landscape,* presenting the initial results from a broad and deep "Census of Scholarly Communication Infrastructure."

Educopia's report[25] makes a number of important calls to action, including the need for a standardized taxonomy of the functional components of scholarly infrastructure. This is a big task. The present landscape analysis will only go a small ways towards providing a common language and framework for talking about scholarly infrastructure as a whole; this is but a baby step toward what is ultimately needed. The Educopia report importantly underscores the challenges projects face in "raising and sustaining appropriate levels of funding to enable them to build and maintain services over time," and, relatedly, the need for "scaled, leveraged efficiencies" to make development sustainable and more risk-tolerant.

To our eyes, the most important call to action made by the Educopia report is for community organization: in "guidance, mentorship, training," in "clarity in their expressions of their purposes and goals," and in the need to bring more stability and predictability to both the technical and financial aspects of infrastructure development.

21 Chodacki, John, Patricia Cruse, Jennifer Lin, Cameron Neylon, Damian Pattinson, and Carly Strasser. "Supporting Research Communications: A Guide," September 2018. https://www.supporters.guide/

From the perspective of our survey of the landscape of open-source publishing projects, the *most important feature is scale*. Almost all of the projects we examined are small—too small to gain critical developer mass as open-source projects (compared, say, to Internet infrastructure projects like Apache or Node.js or the React framework), and too niche or specialized to develop a market-based clientele that might provide meaningful revenue. OJS and Hypothesis are the projects here with the largest scale, but neither is sufficiently either successful or mature to provide a sustainability model for other projects. Most projects are too small, too niche to be sustainable on their own, and will require extrinsic funding sources going forward. But to say that simply shifts the sustainability problem up a level; how does a government or private funding agency continue to fund myriad small projects, with new ones coming onstream all the time?

The lack of scale should not be seen as a failure to grow. Chodacki and colleagues wrote helpfully about the critical importance of trusted relationships in open scholarly communication, and how the emphasis on trust presents challenges for scalability.[26] At the Force2018 conference, Adam Hyde of the Coko Foundation also commented on the need to scale Coko's community slowly enough to maintain a sense of trust among community members.

But inability to scale can mean trouble raising revenue and hence with sustainability over time. There are two common approaches to the problem of scale. One is *consolidation:* let the market shake out so that it supports only a small number of projects that can take the lion's share of available funding and thereby become at least affordable, if not self-sufficient. But this is an unpopular idea, for some obvious reasons. Consolidation like this will squeeze out innovation and adaptability. No one wants a Soviet-style, centrally planned scholarly infrastructure. Similarly, there is considerable concern around the spectre of corporate-style consolidation. Indeed, this is the scenario that led to the idea of community-owned infrastructure in the first place.

The other approach to the problem of scale is coordination and integration—which is what the open ecosystem significantly lacks currently. The opportunity at hand—for funders, for organizers and integrators, and for all actors who would further the overall goal of scholarly community-owned scholarly communication—seems to have come to rest here. How can we build incentives for collaboration and interactivity? How can we encourage, if not technical standardization *per se*, at least standards around APIs and module-level functions? How can we develop financial, governance, and sustainability capacity in the community, so projects have a better long-term footing? At a higher level, how can we leverage the intellectual riches that a plurality of approaches and innovators provides without being mired in a counter-productive environment in which these projects are in competition with each other for users, funding, and a chance to succeed? Competition is well and good, but if the goal is community-owned

22 https://jrost.org/

23 https://osaos.codefor-science.org/

24 https://investinopen.org/

25 Skinner, Katherine. "Mapping the Scholarly Communication Landscape – 2019 Census." Atlanta: Educopia Institute, June 20, 2019. https://educopia.org/2019-census/

infrastructure, competition alone isn't likely to provide it. That scholarly publishing is a classic example of "market failure" is not a new idea.[27]

Concluding thoughts

All of this is to restate the JROST, OSOAS, and IOI agendas, and we welcome new work and new development on these levels. If scale is a structural problem facing many of these projects, then community coordination may go some distance towards addressing it. If longer-term funding for sustainability is needed, then a mediating layer might productively function as a broker of such funding, assuming overhead costs remain low.

We hope this research begins to build a bridge between, on the one hand, thinking about these projects in terms of innovation, features, and interfaces and, on the other hand, the opportunities, and challenges, of supporting community-owned/governed infrastructure. We see a gap between the way we all talk about *projects*—like Manifold, OJS, Editoria, Libero, and so on—and the way we talk about the need for *infrastructure*. The projects do not add up to infrastructure on their own; they are all potential infrastructure components, but have not yet cohered into a comprehensive, networked environment.

In *Roads and Bridges*, Nadia Eghbal offered some reasoned advice for developing effective support strategies for software *as infrastructure*—rather than as product or research tool. Eghbal wrote, "Supporting infrastructure requires embracing the concept of stewardship rather than control."[28] Control is what firms seek in a competitive market, as a means of mitigating risk and consolidating position. If we continue to employ market-informed metaphors and models for these projects—in the idea of competition for funding, for users, for mindshare; in seed funding for innovation as analogous to venture capital; in our *product* focus—we miss the opportunity to make investments in infrastructure qua infrastructure. Eghbal's "roads and bridges" wasn't just a picturesque name; we might add schools and hospitals, and universities.

The key lesson here then might be that layers that support integration, networking, and longer-term sustainability are what need to be funded and developed at this point. If there is a gap it is not software, it's ecosystem integration.

26 Chodacki et al. "Supporting Research Communications," 23.

27 Monograph publishing as an example of the economics term "market failure" was made by Raym Crow in the 2012 AAU-ARL report "A Rational System for Funding Scholarly Monographs." John B Thompson's 2005 *Books in the Digital Age* makes an older but equally detailed exploration of this phenomenon.

Bibliography

Arp, Laurie Gemmill, and Megan Forbes. "It Takes a Village: Open Source Software Sustainability." *LYRASIS*, February 2018. https://doi.org/10.7916/D89G70BS

Bilder, G, Jennifer Lin, and Cameron Neylon. "» Principles for Open Scholarly Infrastructures." *Science in the Open: The Online Home of Cameron Neylon* (blog), February 2015. https://cameronneylon.net/blog/principles-for-open-scholarly-infrastructures/

Chodacki, John, Patricia Cruse, Jennifer Lin, Cameron Neylon, Damian Pattinson, and Carly Strasser. "Supporting Research Communications: A Guide," September 2018. https://www.supporters.guide/

Crow, Raym. "A Rational System for Funding Scholarly Monographs." White Paper. AAU-ARL Task Force on Scholarly Communications, 2012. https://www.arl.org/resources/a-rational-system-for-funding-scholarly-monographs/

Eghbal, Nadia. "Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure." Ford Foundation, July 14, 2016. https://www.fordfoundation.org/about/library/reports-and-studies/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure

Fitzgerald, Brian. "The Transformation of Open Source Software." *MIS Quarterly 30*, no. 3 (2006): 587–598. https://doi.org/10.2307/25148740

Goldstein, Ayala. "Top 10 Open Source Licenses in 2018: Trends and Predictions." *Whitesource*, December 13, 2018. https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions

Joseph, Heather. "Securing Community-Controlled Infrastructure: SPARC's Plan of Action" *College and Research Libraries News 79*, no. 8 (August 2018). https://doi.org/10.5860/crln.79.8.426

Maron, Nancy. "Understanding the Audience of the Public Knowledge Project's Open Source Software." BlueSky to BluePrint, March 2018. https://pkp.sfu.ca/findings-from-community-consultation-2018/

Maxwell, John W., Alessandra Bordini, and Katie Shamash. "Reassembling Scholarly Communications: An Evaluation of the Andrew W. Mellon Foundation's Monograph Initiative (Final Report, May 2016)." *Journal of Electronic Publishing 20*, no. 1 (2017). http://dx.doi.org/10.3998/3336451.0020.101

Schonfeld, Roger C. "Elsevier Acquires Institutional Repository Provider bepress." *The Scholarly Kitchen* (blog), August 2, 2017. https://scholarlykitchen.sspnet.org/2017/08/02/elsevier-acquires-bepress/

Schonfeld, Roger C. "Strategy & Integration Among Workflow Providers." *The Scholarly Kitchen* (blog), November 7, 2017. https://scholarlykitchen.sspnet.org/2017/11/07/strategy-integration-workflow-providers/

28 Eghbal, "Roads and Bridges," 125.

Skinner, Katherine. "Mapping the Scholarly Communication Landscape – 2019 Census." Atlanta: Educopia Institute, June 20, 2019. https://educopia.org/2019-census/

Thompson, John B. *Books In The Digital Age: The Transformation Of Academic And Higher Education Publishing In Britain And The United States*. Polity Press, 2005

Triglav, Jure. "Open Source Collaborative Text Editors." *A Case for Spaceships* (blog), May 7, 2019. https://juretriglav.si/open-source-collaborative-text-editors/

Waters, Donald. "The Monograph Is Dead! Long Live the Monograph!" presented at the *Jisc-CNI Leadership Conference*, July 2, 2018. https://www.slideshare.net/JISC/the-monograph-is-dead-long-live-the-monograph

Watkinson, Charles. "The Academic EBook Ecosystem Reinvigorated: A Perspective from the USA." *Learned Publishing 31*, no. S1 (2018): 280–87. https://doi.org/10.1002/leap.1185

West, Joel, and Siobhán O'Mahony. "The Role of Participation Architecture in Growing Sponsored Open Source Communities." *Industry and Innovation 15*, no. 2 (April 1, 2008): 145–68. https://doi.org/10.1080/13662710801970142

Catalogue of Projects

How to read the Catalogue

Fifty-two projects are catalogued here. The information presented in these entries is a distillation of much more detailed notes. We have attempted to present a short description that explains what the software is and does, followed by basic information about who is behind the project, in terms of leadership, development, funding, and partnership, and some basic info about its lifespan so far.

Following the *Basic Info* section is data pulled from Github and Gitlab repositories. This information is included for illustrative purposes; it is not useful for comparison, for two simple reasons: first, that the projects catalogued here are small enough that quantitative data does not tell an accurate or even compelling story; second, that governance and project management practices of these projects varies widely. As a result, metrics from Github vary much more as a result of how the developers organize themselves than due to actual activity in any given repository. Complicating further is the fact that some projects are in a single respository, and some are spread across many. We have endeavoured to provide usable and credible information here, as evidence of the active status of these projects.

About funding sources

We have listed funding sources for each project as claimed by the projects themselves—either on their websites or in personal communication. Some projects have had many funding sources (not all are current); some project have no funding sources at all beyond the energy and time of their developers.

For the sake of readability in the Catalogue listings, we have used short forms of a number of common funding sources, which we list here in full:

American Mathematical Society (AMS)

Laura and John Arnold Foundation (2 projects)

Canadian Foundation for Innovation (CFI) (2 projects)

Canadian Internet Registration Authority (2 projects)

John Paul Getty Trust (2 projects)

Helmsley Charitable Trust (2 projects)

Howard Hughes Medical Institute (3 projects)

Institute of Museum and Library Services (IMLS) (2 projects)

Knight Foundation (2 projects)

Samuel H. Kress Foundation

MacArthur Foundation (2 projects)

Andrew W Mellon Foundation (14 projects)

Gordon and Betty Moore Foundation (2 projects)

National Endowment for the Humanities (3 projects)

Max Planck Society (3 projects)

Alfred P Sloan Foundation (7 projects)

Shuttleworth Foundation (8 projects)

Siegel Family Foundation

Social Sciences and Humanities Research Council (SSHRC) (2 projects)

Knut and Alice Wallenberg Foundation (3 projects)

Wellcome Trust (3 projects)

dokieli

dokieli is a general-purpose client-side tool for decentralised article publishing, annotations and social interactions based on open Web standards and best practices. dokieli positions itself in a decentralised and interoperable information space where researchers can exercise their autonomy by controlling their identifiers and identities whilst fulfilling the core functions of scientific communication (registration, awareness, certification, archiving).

Basic Info:

Institutional host: dokie.li
URL: https://dokie.li/
Principal investigator: Sarven Capadisli
Contact: info@csarven.ca
Lead developer: Sarven Capadisli
Funding sources: University of Bonn, Massachusetts Institute of Technology, TIB Leibniz Information Centre for Science and Technology, Inrupt Inc.
Development partners:
Partners: Initial release: 2015
Version (as of June 2019): current

Github (as of April 2019):

URL: https://github.com/linkeddata/dokieli/
Language: JavaScript
License: Apache 2.0
Last commit: 2019-04-01
Contributors: 22

Editoria

Editoria is an open-source authoring, editing, and workflow system initially developed by Coko in partnership with the Editoria community underwritten by fiscal sponsor Aspiration Tech and funded by the Mellon Foundation. Editoria is a web-based tool for producing scholarly monographs in both print and ebook forms. Coko's PubSweet framework and Wax editor are underlying technologies in Editoria. Paged.js is available as a print production pathway, as are other format outputs.

Basic Info:

Institutional host: Coko, via Aspiration Tech
URL: https://editoria.pub/
Principal investigator: Allen Gunn (Aspiration Tech),Adam Hyde (Coko)
Contact: alison@coko.foundation
Lead developer: Alexis Georgantas
Funding sources: Mellon, Shuttleworth
Development partners: Coko, PagedMedia
Partners: U California Press; California Digital Library; Aspiration Tech; along with UNC Press; Longleaf Services; Book Sprints; Open Textbook Network, and a growing community of publishers
Initial release: 2017
Version (as of June 2019): Momenvasia (April 2019)

Gitlab (as of April 2019):

URL: https://gitlab.coko.foundation/editoria/editoria
Language: JavaScript
License: MIT
Last commit: 2019-04-23
Contributors: 14

Electric Book

Electric Book is a Jekyll-based tool for producing print PDF, digital PDF, EPUB, website, and app versions of books from a single markdown, YAML, and HTML-based content source. It was developed by consultancy and service provider Electric Book Works.

Basic Info:

Institutional host: Electric Book Works
URL: http://electricbook.works/
Principal investigator: Arthur Attwell
Contact: team@electricbookworks.com
Lead developer: Arthur Attwell
Funding sources: Electric Book Works
Development partners:
Partners: CORE, Shuttleworth Foundation, Oxford University Press, Bettercare, Pan Macmillan
Initial release: 2016
Version (as of June 2019): 0.15

Github (as of April 2019):

URL: https://github.com/electricbookworks/electric-book
Language: JavaScript
License: GPL v3
Last commit: 2019-04-24
Contributors: 5

Enhanced Networked Monographs

Enhanced Networked Monographs (ENM) is an experimental project developed by New York University. It provides a free platform for topic-based and full-text searching on a corpus of books from NYU Press, University of Minnesota Press, and the University of Michigan Press. The platform consists of the ENM search application plus generated topic pages and the customized version of the Topic Curation Toolkit (TCT) used to power/generate them.

Basic Info:

Institutional host: New York University, Digital Library Technology Services of NYU Library
URL: https://wp.nyu.edu/enmproject/
Principal investigator: David Millman
Contact: jonathan.greenberg@nyu.edu
Lead developer: David Arjanik (programmer), Laura Henze (designer)
Funding sources: Mellon
Development partners: Infoloom, Evident Point
Partners: NYU Press; the Digital Library Technology Services department of NYU Libraries; University of Minnesota Press; University of Michigan Press
Initial release: 2018
Version (as of June 2019): ENM search application: v1.1.3

Github (as of April 2019):

URL: https://github.com/NYULibraries/dlts-enm
Language: Go, JavaScript, HTML, CSS, Python
License: Apache 2.0
Last commit: 2019-02-20
Contributors: 2

epub.js

epub.js is a JavaScript library that provides a robust drop-in EPUB reader application to any website, providing styling, pagination, and persistence. The project comes from FuturePress, an offshoot of the UC Berkeley School of Information.

Basic Info:

Institutional host: FuturePress
URL: http://futurepress.org/
Principal investigator: Fred Chasen
Contact: futurepressorg@gmail.com
Lead developer: Fred Chasen
Funding sources: Shuttleworth
Development partners:
Partners:
Initial release: 2014
Version (as of June 2019): 0.3.73

Github (as of April 2019):

URL: https://github.com/futurepress/epub.js
Language: JavaScript
License: MIT
Last commit: 2019-03-29
Contributors: 59

Fidus Writer

Fidus Writer is a web-based, collaborative editor made for academics who need to use citations and/or formulas. Fidus Writer offers a visual editing interface, real-time editing collaboration, a commenting/review workflow system, and a variety of export formats. Fidus provides hosting and styled templates for a monthly fee.

Basic Info:

Institutional host: Lund Info AB
URL: https://www.fiduswriter.org/
Principal investigator: Johannes Wilm
Contact: mail@fiduswriter.org
Lead developer: Johannes Wilm
Funding sources: Startup Chile (CORFO); German Research Foundation, hosting services
Development partners: Opening Scholarly Communications in the Social Sciences (OSCOSS, University of Bonn); GESIS (Leibniz Institute for the Social Sciences)
Partners:
Initial release: 2013
Version (as of June 2019): 3.6.5

Github (as of April 2019):

URL: https://github.com/fiduswriter/fiduswriter
Language: JavaScript, Python
License: AGPL
Last commit: 2019-04-15
Contributors: 20

Fulcrum

Fulcrum is the University of Michigan Library's ebook hosting, preservation, and media integration platform, devloped on top of the Samvera repository platform. Fulcrum allows authors and publishers to integrate multimedia elements into a book—linked from a print book or directly integrated in an ebook—while providing a robust, richly described, and accessible reader environment and a discoverability platform for ebook collections. Fulcrum is a platform available to UMichigan Press authors, as well as a service offered to other publishers. Fulcrum makes use of epub.js, AblePlayer, Hypothes.is, and Editoria (in testing) to provide basic and enhanced functionality.

Basic Info:

Institutional host: University of Michigan Library and Press
URL: http://fulcrum.org
Principal investigator: Charles Watkinson
Contact: fulcrum-info@umich.edu
Lead developer: Jeremy Morse
Funding sources: Mellon
Development partners:
Partners: ACLS Humanities E-Book, Northwestern U Press, Penn State, U Minnesota Press, Lever Press, NYU Press, U Michigan Press, Indiana U Press, Amherst College Press, University of Sussex Library, National Museum of Japanese History.
Initial release: 2018
Version (as of June 2019): 2.38 (Heliotrope)

Github (as of April 2019):

URL: https://github.com/mlibrary/heliotrope
Language: Ruby
License: Apache 2.0
Last commit: 2019-04-23
Contributors: 15

Grobid

GROBID (or Grobid) stands for GeneRation Of BIbliographic Data. It is a machine-learning library for extracting, parsing, and re-structuring journal articles in PDF format into structured TEI-encoded documents that can then be transformed to JATS XML. Grobid represents a best-of-breed example (see https://arxiv.org/abs/1802.01168) of the shift from traditional parser-based approaches to machine-learning models for converting legacy documents to XML. Grobid is employed in the PKP's Open Typesetting Stack.

Basic Info:

Institutional host: independent
URL: https://grobid.readthedocs.io/en/latest/Introduction/
Principal investigator: Patrice Lopez
Contact: patrice.lopez@science-miner.com
Lead developer: Patrice Lopez
Funding sources:
Development partners: various
Partners:
Initial release: 2011
Version (as of June 2019): 0.5.4

Github (as of April 2019):

URL: https://github.com/kermitt2/grobid
Language: Java
License: Apache 2.0
Last commit: 2018-04-25
Contributors: 28

HIRMEOS OA Metrics

Born out of the OPERAS project: European Research Infrastructure for the development of open scholarly communication in the social sciences and humanities; HIRMEOS seeks to build functionality for research monographs in the European open-science infrastructure. This metrics project normalizes book identifiers (ISBNs, DOIs), provides modular "drivers" to gather various metrics (Google Analytics, JSTOR, COUNTER, etc.) and altmetrics (social media sources), and then aggregates these so publishers have access to usage and traffic data on ebooks. The usage data code has been developed by the UK-based Open Book Publishers. Altmetrics code has been developed by Ubiquity Press.

Basic Info:

Institutional host: OPERAS
URL: https://metrics.operas-eu.org/docs/getting-started
Principal investigator: Pierre Mounier
Contact: javi@openbookpublishers.com
Lead Developers: Javier Arias; Rowan Hatherley
Funding sources: EU Horizon 2020
Development partners: Open Book Publishers, Ubiquity Press
Partners: CNRF, NHRF - EIE, OAPEN, Max Weber Stiftung, UGOE, DAIRIAH ERIC, UNITO
Initial release: 2018
Version (as of June 2019): current

Github (as of April 2019):

URL: https://github.com/hirmeos
Language: Python
License: MIT
Last commit: 2019-04-23
Contributors: 2

Hy-phen

Hy-phen is a JavaScript implementation of Francis Liang's TeX hyphenation algorithm.

Basic Info:

Institutional host: independent
URL: https://github.com/ytiurin/hyphen
Principal investigator: Eugene Tiurin
Contact:
Lead developer: Eugene Tiurin
Funding sources:
Development partners:
Partners:
Initial release: 2016
Version (as of June 2019): 1.1.1

Github (as of April 2019):

URL: https://github.com/ytiurin/hyphen
Language: JavaScript
License: MIT
Last commit: 2019-03-20
Contributors: 7

Hyphenopoly

Hyphenopoly is a JavaScript library for providing robust hyphenation in HTML, especially while hyphenation remains patchily supported by web browsers, especially across multiple languages. Hyphenopoly provides hyphenation dictionaries and algorithms derived from Francis M Liang's classic TeX hyphenation algorithm. Hyphenopoly can be dropped in to any website. Hyphenopoly supercedes an earlier JS system Hyphenator.

Basic Info:

Institutional host: independent
URL: http://mnater.github.io/Hyphenopoly/
Principal investigator: Mathias Nater
Contact: mathiasnater@gmail.com
Lead developer: Mathias Nater
Funding sources:
Development partners:
Partners:
Initial release: 2018
Version (as of June 2019): 3.0.2

Github (as of April 2019):

URL: https://github.com/mnater/Hyphenopoly
Language: JavaScript
License: MIT
Last commit: 2019-04-04
Contributors: 5

Hypher

Hypher is a hyphenation engine written in JavaScript for web browsers using jQuery. It comes with hyphenation dictionaries for more than 30 languages.

Basic Info:

Institutional host: independent
URL: http://www.bramstein.com/working/
Principal investigator: Bram Stein
Contact: b.l.stein@gmail.com
Lead developer: Bram Stein
Funding sources:
Development partners:
Partners:
Initial release: 2012
Version (as of June 2019): 2.5 (2016)

Github (as of Aug 2018):

URL: https://github.com/bramstein/hypher
Language: JavaScript
License: BSD
Last commit: 2018-07-29
Contributors: 10

Hypothesis

Hypothesis is a general-purpose web annotation platform that enables users to annotate any text on the Internet, including HTML, EPUBs, text/CSV files, and online or downloaded PDFs. Users can highlight text, add their comments and post those publicly, privately, or in the context of private or public groups. They can also reply to or share annotations, each of which is available at a unique URL. Hypothesis is centrally hosted, offers a robust API, and integrates with most popular publishing and educational systems and can be added to any website with a single line of JavaScript.

Basic Info:

Institutional host: Hypothesis Project (nonprofit)
URL: https://web.hypothes.is/
Principal investigator: Dan Whaley
Contact: https://web.hypothes.is/contact/
Lead developer: Lyza Danger Gardner; Sean Hammond; Robert Knight; Hannah Stepanek
Funding sources: Helmsley, Knight, Mellon, Omidyar, Shuttleworth, Sloan, Schmidt Futures, MDPI.
Development partners:
Partners: AAAS; Cambridge University Press; The Johns Hopkins University Press; Michigan Publshing; NYU, OJS, eLife, and many more https://web.hypothes.is/partners/
Initial release: 2011
Version (as of June 2019): 1.161 (browser app)

Github (as of April 2019):

URL: https://github.com/hypothesis/h
Language: Python
License: BSD
Last commit: 2019-04-17
Contributors: 54

Janeway

Janeway is journal management software developed by the Birkbeck Centre for Technology and Publishing for the Open Library of Humanities (OLH) at Birkbeck, University of London. Janeway integrates Crossref, iThenticate, Portico, and CLOCKSS services to provide a full-featured OA journal publishing platform. Janeway is a Django-based web application.

Basic Info:

Institutional host: Centre for Technology and Publishing at Birkbeck
URL: https://janeway.systems/
Principal investigator: Martin Paul Eve
Contact: martin.eve@bbk.ac.uk
Lead developer: Andy Byers, Mauro Sanches
Funding sources: donors, clients
Development partners: Carnegie Mellon University Libraries, California Digital Library
Partners: University of Iowa Digital Press, University of Huddersfield Press
Initial release: 2017
Version (as of June 2019): 1.3.5.1

Github (as of April 2019):

URL: https://github.com/BirkbeckCTP/janeway
Language: Python
License: AGPL v3
Last commit: 2019-04-24
Contributors: 10

Jupiter Notebook

Jupyter Notebook is a web-based notebook environment for interactive computing, part of the mutifacted Project Jupyter (formerly iPython) which seeks to provide an open platform and toolkit for interactive and reproducible computing. It is a browser-based application that facilitates creation and sharing of documents that contain live code (over 40 programming languages), equations, visualizations, and narrative text. Jupyter Hub is a multi-user version for classrooms and labs. JupyterLab, released in 2018, provides a modern web-based user interface for Jupytern Notebooks. Jupyter's file format is a JSON document with hooks to interactive runtime kernels for specific languages and connections to big data sources.

Basic Info:

Institutional host: NumFOCUS
URL: http://jupyter.org/
Principal investigator: Jupyter Steering Council
Contact: project.jupyter@gmail.com
Lead developer: Jupyter Steering Council
Funding sources: Helmsley, Sloan, Moore, Google, Microsoft, rackspace, fastly, Quansight, Schmidt Futures, European Union Funding for Research and Innovations
Development partners:
Partners: Anaconda, Bloomberg, Netflix, Cal Poly, UC Berkeley, QuantStack, TwoSigma, JPMorgan Chase, UC Merced, Amazon Web Services
Initial release: 2011
Version (as of June 2019): current

Github (as of April 2019):

URL: https://github.com/jupyter (and many more repos)
Language: JavaScript, Python
License: BSD
Last commit: 2019-04-30
Contributors: 400+

KaTeX

KaTeX is a LaTeX-based typesetting tool for mathematical expressions developed by the Khan Academy. It is billed as the fastest math typesetting library for the web because it renders math in real time without the need to reflow the page. It is self-contained with no dependencies and can run server-side or in the browser.

Basic Info:

Institutional host: Khan Academy
URL: https://katex.org/
Principal investigator: Erik Demaine
Contact: opensource@khanacademy.org
Lead developer: Emily Eisenberg, Sophie Alpert, Kevin Barabash
Funding sources: Khan Academy
Development partners:
Partners:
Initial release: 2016
Version (as of June 2019): 0.10.2

Github (as of April 2019):

URL: https://github.com/KaTeX/KaTeX
Language: JavaScript
License: MIT
Last commit: 2019-04-23
Contributors: 86

Lens

Lens is an online article-reading environment developed by eLife that—by treating a JATS journal article as a database—makes it possible to explore figures, figure descriptions, references and more without losing one's place in the article text. Lens was designed using the Substance libraries. Much of its functionality is now in eLife's Libero Producer tool.

Basic Info:

Institutional host: eLife
URL: https://lens.elifesciences.org/about/#info/all
Principal investigator: Ivan Grubisic
Contact: https://github.com/ivangrub
Lead developer: Ivan Grubisic
Funding sources: eLife
Development partners: Substance
Partners: Fidus Writer; Substance; Hyopthes.is
Initial release: 2013
Version (as of June 2019): 2.0.0

Github (as of April 2019):

URL: https://github.com/elifesciences/lens
Language: JavaScript
License: BSD
Last commit: 2018-01-08
Contributors: 12

le-tex Transpect

le-tex Transpect is an XProc- and XSLT-based framework and suite of modules for managing, schema checking, and converting from/to XML-based formats such as .docx, IDML, EPUB, HTML, DocBook, TEI and JATS. le-tex Transpect also provides a framework for combining modules into publishing workflows with revision control and custom, cascade-based configuration. le-tex Transpect can run standalone or integrated into publishing workflows. A simple upload interface and an HTTP API is available, as is hosted operation and maintenance agreements for professional use.

Basic Info:

Institutional host: le-tex Publishing Services
URL: https://transpect.github.io/
Principal investigator: Martin Kraetke, Gerrit Imsieke
Contact: letexml@le-tex.de
Lead developer: Martin Kraetke, Gerrit Imsieke
Funding sources: le-tex publishing services GmbH and various customers
Development partners:
Partners:
Initial release: 2013
Version (as of June 2019): current

Github (as of April 2019):

URL: https://github.com/transpect
Language: XSLT, XProc
License: BSD2
Last commit: 2019-04-22
Contributors: 12

Libero Producer

Libero Producer is the first of three journal publishing modules developed by eLife. Libero Producer is based on Substance.io's Texture editor which provides a visual, browser-based JATS XML editing and viewing interface.

Basic Info:

Institutional host: eLife
URL: https://libero.pub/
Principal investigator: Maël Plaine
Contact: hello@libero.pub
Lead developer: Maël Plaine
Funding sources: Howard Hughes Medical Institute, Max Planck Society, Wellcome Trust, Knut and Alice Wallenberg Foundation
Development partners: Digirati
Partners: Coko; Substance, Hindawi;
Initial release: 2019
Version (as of June 2019): current

Github (as of June 2019):

Libero Producer is tightly tied to Texture editor; see the Texture repo at https://github.com/substance/texture

Libero Publisher

Libero Publisher is the third of three journal publishing modules developed by eLife. Libero Publisher provides post-production hosting, publication, and journal management functions, including dashboards, ElasticSearch, and APIs for third-party integration.

Basic Info:

Institutional host: eLife
URL: https://libero.pub/
Principal investigator: Maël Plaine
Contact: hello@libero.pub
Lead developer: Maël Plaine
Funding sources: Howard Hughes Medical Institute, Max Planck Society, Wellcome Trust, Knut and Alice Wallenberg Foundation
Development partners: Digirati
Partners: Coko; Substance, Hindawi;
Initial release: 2019
Version (as of June 2019):

Github (as of June 2019):

URL: https://github.com/libero
Language: PHP
License: MIT
Last commit: 2018-11-09
Contributors: 5

Libero Reviewer

Libero Reviewer is the second of three journal publishing modules developed by eLife. Libero Reviewer handles article submission and peer review workflow management. It is built on the Coko Foundation's PubSweet framework and was designed in collaboration with Coko and Hindawi.

Basic Info:

Institutional host: eLife
URL: https://libero.pub/
Principal investigator: Maël Plaine
Contact: hello@libero.pub
Lead developer: Maël Plaine
Funding sources: Howard Hughes Medical Institute, Max Planck Society, Wellcome Trust, Knut and Alice Wallenberg Foundation
Development partners: Digirati
Partners: Coko; Substance, Hindawi;
Initial release: 2019
Version (as of June 2019): current

Github (as of June 2019):

URL: https://github.com/elifesciences/elife-xpub
Language: JavaScript
License: MIT
Last commit: 2019-06-21
Contributors: 18

Lodel

Lodel is the journal publishing software for the French OpenEdition publishing platform. It provides content management and import/conversion to bring word processor documents into an XML-based article production environment.

Basic Info:

Institutional host: OpenEdition, CNRS France
URL: http://www.lodel.org/index.html
Principal investigator: Raphaëlle Daudé
Contact: lodel@lodel.org
Lead developer: Raphaëlle Daudé
Funding sources: OpenEdition, Centre national de la recherche scientifique (CNRS)
Development partners:
Partners:
Initial release: 2006
Version (as of June 2019): 1.03

Github (as of April 2019):

URL: https://github.com/OpenEdition/lodel
Language: PHP
License: GPL v2
Last commit: 2019-04-23
Contributors: 11

Manifold Scholarship

Manifold is a collaborative, web-based scholarly publishing system designed by the University of Minnesota Press and the CUNY Graduate Center. Manifold provides a dynamic approach to publishing book-length works capable of gathering commentary, annotation, and revisions within the publication. Built to publish long-form digital monographs, Manifold is also used in service of open educational resources, journals, and collaborative scholarly projects. Tt is currently used by twenty-eight publishers, including the University of Minnesota Press, the City University of New York, and the University of Arizona Press, as well as digital humanities centers and teaching and learning centers.

Basic Info:

Institutional host: University of Minnesota Press & the CUNY Graduate Centre
URL: https://manifoldapp.org/
Principal investigator: Doug Armato and Matthew K Gold
Contact: contact@manifoldapp.org
Lead developer: Zach Davis
Funding sources: Mellon
Development partners: Cast Iron Coding
Partners: CUNY Graduate Centre Digital Scholarship Lab, plus 20 pilot-test presses
Initial release: 2018
Version (as of June 2019): 3.0.1

Github (as of April 2019):

URL: https://github.com/ManifoldScholar/manifold
Language: Ruby, JavaScript
License: GPL v3
Last commit: 2019-04-23
Contributors: 10

MathJax

MathJax is a JavaScript display engine for mathematics typesetting that works in web browsers. It provides support for LaTeX, MathML, and AsciiMath in the web based interace. MathJax has a modular design; it is designed for accessibility and interoperability with other applications.

Basic Info:

Institutional host: NumFOCUS
URL: https://www.mathjax.org/
Principal investigator: Davide Cervone; Volker Sorge
Contact: info@mathjax.org
Lead developer: Davide Cervone
Funding sources: American Mathematical Society (AMS); Society for Industrial and Applied Mathematics (SIAM); IEEE; Elsevier; and a list of "supporters"
Development partners:
Partners: see Funding Source above
Initial release: 2010
Version (as of June 2019): v3 beta

Github (as of April 2019):

URL: https://github.com/mathjax/MathJax
Language: JavaScript
License: Apache 2.0
Last commit: 2018-07-19
Contributors: 29

Mukurtu

Mukurtu is a content management system developed by Washington State University to serve as a repository for Indigenous communities to manage, share, and exchange their digital heritage in culturally relevant and ethically minded ways. Mukurtu has innovated significantly in developing access-oriented metadata that goes beyond typical OA ideals to support fine-grained Traditional Knowledge access protocols.

Basic Info:

Institutional host: Washington State University, Center for Digital Scholarship and Curation
URL: http://mukurtu.org/
Principal investigator: Dr Kimberly Christen
Contact: support@mukurtu.org
Lead developer: Steve Taylor
Funding sources: Washington State U Foundation; NEH; IMLS; Feltzer Institute; WIPO; Mellon
Development partners: Kanopi Studios
Partners:
Initial release: 2012
Version (as of June 2019): 2.1.2

Github (as of April 2019):

URL: https://github.com/MukurtuCMS/mukurtucms
Language: PHP
License: GPL v2
Last commit: 2019-04-17
Contributors: 3

Omeka

Omeka is a web-based platform for creating and sharing digital collections and creating media-rich online exhibits. Initially developed at George Mason University and sustained by the Corporation for Digital Scholarship, Omeka has been primarily targeted towards libraries, museums, historical societies, and the like. Omeka enables institutions to publish collections and narrative exhibits to the web easily, but its publishing features, standards-based metadata, collection management, and authoring tools make it a publishing system more generally. Omeka S, a newer variant than Omeka Classic, supports multiple publications from a single installation, with a linked open data infrastructure.

Basic Info:

Institutional host: George Mason University
URL: https://omeka.org/
Principal investigator: Sharon Leon
Contact: outreach@omeka.org
Lead developer: John Flatness
Funding sources: Sloan; Kress; Mellon; Getty; IMLS; NEH; Library of COngress Corporation for Digital Scholarship
Development partners: Corporation for Digital Scholarship
Initial release: 2008
Version (as of June 2019): Omeka S 1.4; Omeka Classic 2.7

Github (as of April 2019):

Omeka Classic

URL: https://github.com/omeka/Omeka
Language: PHP
License: GPL v3
Last commit: 2019-03-27
Contributors: 32

Omeka S

URL: https://github.com/omeka/omeka-s
Language: PHP
License: GPL v3
Last commit: 2019-04-22
Contributors: 19

Open Journal Systems

Open Journal Systems (OJS) is the world's most widely used open-source journal management and publishing system. Developed by the Public Knowledge Project (PKP), OJS can be downloaded and installed locally but is also commonly hosted by library or institutional IT services. OJS manages workflow for the entire refereed publishing process, providing a common model for the operational processes of a peer-reviewed journal. Through the PKP, OJS also connects with myriad indexing, identification, discoverability, and preservation services.

Basic info:

Institutional host: Public Knowledge Project, SFU
URL: https://pkp.sfu.ca/ojs
Principal investigator: John Willinsky
Contact: kstranac@sfu.ca
Lead developer: Alec Smecher
Fundering sources: CFI; SSHRC; CIRA; Arnold; MacArthur; SFU Library; Stanford University MediaX; as well as a network of fee-for-service "sustainers"
Development partners: Ontario Council of University Libraries (OCUL), U of Alberta Libraries, UBC Libraries, U of Pittsburgh Libraries, Ubiquity Press.
Partners: 24 "strategic partners"
Initial release: 2002
Version (as of June 2019): 3.1.2

Github (as of April 2019):

URL: https://github.com/pkp/ojs, https://github.com/pkp/pkp-lib
Language: PHP
License: GPL v2
Last commit: 2019-04-24
Contributors: 92

Open Monograph Press

Open Monograph Press (OMP) is a book-oriented workflow manager and online publishing platform. Developed by the Public Knowledge Project, it shares its codebase with Open Journal Systems. OMP can handle monographs and edited volumes with multiple authors, as well as manage author submissions, editor assignments, reviewers, indexers, and others in book production. OMP is one of very few open-source tools that produce the trade-industry standard ONIX metadata. Its public-facing side can feature thumbnail covers in a vatalog view, as well as Spotlight marketing features.

Basic Info:

Institutional host: Public Knowledge Project, SFU
URL: https://pkp.sfu.ca/omp
Principal investigator: John Willinsky
Contact: kstranac@sfu.ca
Lead developer: Alec Smecher
Funding sources: CFI; SSHRC; CIRA; Arnold; MacArthur; SFU Library; Stanford University MediaX; as well as a network of fee-for-service "sustainers"
Development partners:
Partners:
Initial release: 2011
Version (as of June 2019): 3.1.2

Github (as of April 2019):

URL: https://github.com/pkp/omp, https://github.com/pkp/pkp-lib
Language: PHP
License: GPL v2
Last commit: 2019-04-17
Contributors: 30

Open Typesetting Stack

Open Typesetting Stack (OTS) is an article conversion/ingest service developed by the Public Knowledge Project to convert word-processor and PDF versions of articles into JATS XML for publication. OTS integrates a host of other parsing and conversion tools (including the machine-learning tool Grobid) and external services to provide the most accurate possible XML without additional user input. This service—and its OJS plugin integration—is intended to decrease the labour involved in production, and to facilitate the creation of archive-friendly and web-native article formats. OTS is in maintenance mode as of this writing.

Basic Info:

Institutional host: Public Knowledge Project, SFU
URL: https://pkp.sfu.ca/open-typesetting-stack/
Principal investigator: Alex Garnett
Contact: axfelix@gmail.com
Lead developer: Alex Garnett
Funding sources: Stanford University MediaX; Canadian Internet Registration Authority; Open Library of the Humanities
Development partners:
Partners:
Initial release: 2015
Version (as of June 2019): current as of Aug 2018

Github (as of April 2019):

URL: https://github.com/pkp/ots
Language: JavaScript
License: GPL v3
Last commit: 2018-08-04
Contributors: 8

paged.js

Paged.js is a comprehensive print-oriented production system that runs on CSS and JavaScript in a web browser. Developed by the PagedMedia initiative, it aims to offer a best-of-breed CSS-based typesetter as open-source software. It can display both paginated output and editable CSS on a page so that the CSS can be tweaked and changes can be viewed in real time.

Basic Info:

Institutional host: Cabbage Tree Labs
URL: https://www.pagedmedia.org/paged.js
Principal investigator: Adam Hyde
Contact: adam@booksprints.net
Lead developer: Fred Chasen
Funding sources: Shuttleworth
Development partners: Editoria, Coko
Partners: Editoria; C&F Editions
Initial release: 2018
Version (as of June 2019): 1.3.4

Gitlab (as of April 2019):

URL: https://gitlab.pagedmedia.org/tools/pagedjs
Language: JavaScript
License: MIT
Last commit: 2019-04-03
Contributors: 10

Pandoc

Pandoc is a robust, multi-format document conversion tool that can read from and write to a vast number of file formats. Pandoc can work with a range of markup formats, markdown, word-processor files, and it supports integration with tools like LaTeX and reference managers, as well as a host of web-based formats. Several different input and exports formats for math are handled, including MathJax, LaTeX, and translation to MathML. Pandoc also includes a powerful system for automatic citations and bibliographies. Pandoc is usable as a command-line tool as well as an integrated library, and is used in several other publishing toolkits.

Basic Info:

Institutional host: independent
URL: https://pandoc.org/
Principal investigator: John MacFarlane
Contact: pandoc-discuss@googlegroups.com
Lead developer: John MacFarlane
Funding sources:
Development partners:
Partners:
Initial release: 2007
Version (as of June 2019): 2.7.3

Github (as of April 2019):

URL: https://github.com/jgm/pandoc
Language: Haskell
License: GPL v2
Last commit: 2019-04-23
Contributors: 268

Paperbuzz

Paperbuzz is a tool that calculates metrics from Crossref Event Data: sharing, linking, and referencing articles online. Paperbuzz is developed and maintained by Our Research with the support of the Public Knowledge Project (PKP). Paperbuzz offers an API that is used by PaperbuzzViz, a JavaScript library to visualize the metrics and by the Paperbuzz OJS Plugin that brings these visualization to OJS article pages, both of which are developed and maintained by PKP.

Basic Info:

Institutional host: Public Knowledge Project, SFU
URL: https://www.paperbuzz.org/
Principal investigator: Juan Alperin
Contact: team@ourresearch.org
Lead developer: Juan Alperin
Funding sources: CO.SHS/CFI
Development partners: Our Research (formerly ImpactStory)
Partners:
Initial release: 2019
Version (as of June 2019): 1.0.0

Github (as of Aug 2018):

URL: https://github.com/Impactstory/paperbuzz-api https://github.com/jalperin/paperbuzzviz/
Language: JavaScript
License: MIT **Contributors:** 3

Phenom Reviewer

Phenom Reviewer is Hindawi’s article submission and editorial workflow module. It is built on the Coko Foundation’s PubSweet framework, and is designed in collaboration with Coko and eLife. Phenom Reviewer is part of a larger suite of tools in early development, which will comprise “Producer” and “Publisher” modules similar to eLife’s Libero suite.

Basic Info:

Institutional host: Hindawi
URL: https://demo.review.hindawi.com
Principal investigator: Andrew Smeall
Contact: andrew.smeall@hindawi.com
Lead developer: Bogdan Cochior
Funding sources: Hindawi
Development partners: Coko, eLife;
Partners:
Initial release: September 2018
Version (as of June 2019): 2.31

Gitlab (as of July 2019):

URL: https://gitlab.com/hindawi/xpub/xpub-review
Language: JavaScript
License: MIT
Last commit: 2019-07-01
Contributors: 17

Phenom Screener

Phenom Screener is Hindawi's module that performs ethical and technical checks on article submissions including plagiarism screening, identity verification, materials checking, and fraud prevention.

Basic Info:

Institutional host: Hindawi
URL: https://demo.review.hindawi.com
Principal investigator: Andrew Smeall
Contact: andrew.smeall@hindawi.com
Lead developer: Bogdan Cochior
Funding sources: Hindawi
Development partners: Coko, eLife
Partners:
Initial release: May 2019
Version (as of June 2019):

Gitlab (as of July 2019):

URL: https://gitlab.com/hindawi/xpub/xpub-screening
Language: JavaScript
License: MIT
Last commit: 2019-07-02
Contributors: 7

Pressbooks

Pressbooks is a web-based book editing and production system that exports in multiple formats: ebooks, webbooks, print-ready PDF, and various XML types. The system is built on top of Wordpress, but makes significant changes to the admin interface, presentation layer, and export routines to for web, ebook, and print formats. Pressbooks is widely used in the open textbook and open educational resouces community.

Basic Info:

Institutional host: Pressbooks
URL: https://pressbooks.com/
Principal investigator: Hugh McGuire
Contact: hugh@rebus.foundation
Lead developer: Ned Zimmerman
Funding sources: client-supported
Development partners: Bight.ca
Partners: Rebus Foundation
Initial release: 2011
Version (as of June 2019): 5.8.0

Github (as of April 2019):

URL: https://github.com/pressbooks/pressbooks/
Language: PHP
License: GPL v3
Last commit: 2019-04-22
Contributors: 32

ProseMirror

ProseMirror is a JavaScript framework to develop visual text editors online. It can support collaborative editing in real time. It has a modular architecture that makes sure users only load the code they need, and can replace parts of the system as needed. ProseMirror supports extensible document schemas that allow users to edit documents with a custom structure without writing their own editor from scratch. It has a plugin system that allows users to easily enable additional functionality, and package their own extensions in a convenient format. Prosemirror is used by several major online news sources (NYTimes, Guardian), as well as inside tools like PubPub and Coko Foundation's Wax editor.

Basic Info:

Institutional host: independent
URL: http://prosemirror.net/
Principal investigator: Adrian Heine né Lang, Marijn Haverbeke
Contact: mail@adrianheine.de; marijnh@gmail.com
Lead developer: Marijn Haverbeke
Funding sources: Shuttleworth; crowdsourcing
Development partners:
Partners:
Initial release: 2017
Version (as of June 2019): 1.9.6 (prosemirror-view)

Github (as of June 2019):

URL: https://github.com/ProseMirror/prosemirror
Language: JavaScript
License: MIT
Last commit: 2019-05-28
Contributors: 6

PubPub

PubPub is an online authoring and publishing platform developed by MIT Press and the MIT Knowledge Futures Group. It supports community-based collaborative drafting, review, and publication of scholarly work "using an integrated and iterative process." It supports journals, books, lab communications and events. PubPub is designed to be centrally hosted, and PubPub provides publishing services as part of a tiered-price hosting packages.

Basic Info:

Institutional host: Knowledge Futures Group @ MIT
URL: https://pubpub.org
Principal investigator: Travis Rich
Contact: Catherine Ahearn, team@pubpub.org
Lead developer: Travis Rich
Funding sources: Joi Ito; Reid Hoffman; Siegel; Knight; MacArthur; Sloan
Development partners:
Partners: MIT Media Lab, client publishers
Initial release: 2017
Version (as of June 2019): 6.0.0

Github (as of April 2019):

URL: https://github.com/pubpub/pubpub
Language: JavaScript
License: GPL v2
Last commit: 2019-04-02
Contributors: 10

PubSweet

PubSweet is a foundational system developed by Coko as a “component-based framework” upon which to build publishing tools. PubSweet is a simple but flexible way to adapt to different kinds of system needs. For instance, both the book-oriented Editoria and the journal-oriented Libero Reviewer are built on PubSweet foundations. PubSweet’s community includes Hindawi, eLife, Wormbase, Digital Science, and the EBI’s Europe PMC Plus platform.

Basic Info:

Institutional host: Coko
URL: https://coko.foundation/
Principal investigator: Adam Hyde
Contact: team@coko.foundation
Lead developer: Jure Triglav
Funding sources: Shuttleworth, Hindawi, Arnold, Sloan, Moore, Mellon
Development partners:
Partners: EuropePMC, Hindawi, eLife, WormBase
Initial release: 2017
Version (as of June 2019): current

Gitlab (as of June 2019):

URL: https://gitlab.coko.foundation/pubsweet/pubsweet
Language: JavaScript
License: MIT
Last commit: 2019-06-21
Contributors: 48

Quire

Quire is a book production tool developed by the J Paul Getty Trust. It is a multiformat publishing framework that can create digital and print books, such as museum and gallery exhibition catalogues, collected volumes, and scholarly monographs. Quire is designed around the Hugo static-site generator tool, which can compile and export books, working from markdown source. Quire has extensive support for media, including rich image metadata handling. It is currently without an explicit open-source license.

Basic Info:

Institutional host: Getty Museum
URL: https://github.com/gettypubs/quire
Principal investigator: Greg Albers
Contact: galbers@getty.edu
Lead developer: Matthew Hrudka
Funding sources: Getty
Development partners:
Partners:
Initial release: 2017
Version (as of June 2019): Alpha

Github (as of April 2019):

URL: https://github.com/gettypubs/quire
Language: Go
License: *currently in private beta; access by request*
Last commit: 2019-03-27
Contributors: 4

Readium

Readium provides a “set of software building blocks” for the development of standardized EPUB and web publication reader applications for a variety of contexts—browser-based, mobile app, and desktop. Readium is a set of libraries and frameworks, and also a foundation and international community dedicated to ebook implementation standards.

Basic Info:

Institutional host: Readium Foundation
URL: https://readium.org/
Principal investigator: Hadrien Gardeur
Contact: contact@edrlab.org
Lead developer: Hadrien Gardeur
Funding sources: members
Development partners: European Digital Reading Lab (EDRLab)
Partners: Members include: Editis, Hachette, Madrigall, Media Participations, Syndicat National de l’édition, Cercle de la Librarie, Centre National du livre, the French State and Cap Digital.
Initial release: 2012
Version (as of June 2019): R2

Github (as of June 2019):

URL: https://github.com/readium/readium-desktop
Language: TypeScript
License: BSD
Last commit: 2019-06-20
Contributors: 7

Rebus Ink

Rebus Ink is a web-based digital reading application built to help scholars construct arguments. It's a personal, online workspace that lets you do more with digital texts, focusing on scholarly reading and research, note-taking, citations, and collections management. Rebus Ink is built on open principles: open source, open web, open APIs, with a focus on user-data portability and privacy.

Basic Info:

Institutional host: Rebus Foundation
URL: https://rebus.ink/
Principal investigator: Hugh McGuire
Contact: hugh@rebus.foundation
Lead developer: Baldur Bjarnasson
Funding sources: Mellon
Development partners:
Partners: Partners: Hypothes.is, Michigan Library/Fulcrum, University of Minnesota Press, MIT Press, University of California Press, University of Guelph Libraries, ACLS Humanities E-Book, UC Davis Library
Initial release: 2019
Version (as of June 2019): Alpha

Github (as of June 2019):

URL: https://github.com/RebusFoundation/reader-api
Language: JavaScript
License: AGPL
Last commit: 2019-06-20
Contributors: 4

Rua

Rua is a book publishing workflow management application developed by Ubiquity Press and is “designed to assist with the monograph publishing life cycle” from proposal to publication. Rua forms the core of the Ubiquity Book Manager service. Rua is designed around the Django framework.

Basic Info:

Institutional host: Ubiquity Press
URL: https://github.com/ubiquitypress/rua
Principal investigator: Brian Hole
Contact: https://www.ubiquitypress.com/site/contact/
Lead developer: Stuart Jennings
Funding sources: Ubiquity Press
Development partners:
Partners:
Initial release: 2015
Version (as of June 2019): 3.1.8 alpha

Github (as of April 2019):

URL: https://github.com/ubiquitypress/rua
Language: Python
License: GPL v2
Last commit: 2019-04-02
Contributors: 10

Scalar

Scalar is a multimedia authoring and publishing platform developed by the Alliance for Networking Visual Culture at University of Southern California. Scalar is designed for long-form, digital native scholarly research. Scalar enables users to assemble media from multiple sources and juxtapose them with text in a variety of ways with minimal technical expertise required. The platform also supports collaborative authoring workflows and reader commentary.

Basic Info:

Institutional host: USC, Alliance for Networking Visual Culture
URL: https://scalar.me/anvc/
Principal investigator: Tara McPherson
Contact: alliance4nvc@gmail.com
Lead developer: Craig Dietrich
Funding sources: Mellon, NEH
Development partners:
Partners: Shoah Foundation Institute, Critical Commons, the Hemispheric Institute's Digital Video Library, and the Internet Archive, Getty Library, Duke University Press, MIT Press, NYU Press, Open Humanities Press, U. of California Press, U. of Michigan Press
Initial release: 2013
Version (as of June 2019): 2.5.2

Github (as of April 2019):

URL: https://github.com/anvc/scalar
Language: PHP
License: ECL 2.0
Last commit: 2019-04-21
Contributors: 15

Shiny

Shiny is an authoring and editorial development software developed by RStudio. It allows users to interact with web-based interactive applications that contain data and analysis using R. Shiny can create standalone apps on a webpage or embed them in R Markdown documents or build dashboards. Shiny requires only a R installation and a web browser.

Basic Info:

Institutional host: RStudio
URL: https://shiny.rstudio.com
Principal investigator:
Contact: info@rstudio.com
Lead developer: Joe Cheng
Funding sources: RStudio
Development partners:
Partners:
Initial release: 2017
Version (as of June 2019): 1.3.2

Github (as of April 2019):

URL: https://github.com/rstudio/shiny
Language: R
License: GPL v3
Last commit: 2019-04-23
Contributors: 42

Stencila

Stencila is an authoring and editorial development software developed by Code for Science & Society. It provides an integrated word processor, coding (R, Python, and SQL), and spreadsheet interface in the browser, and the resulting interactive document (using the same file format used by the Texture editor, with which Stencila shares code) is shareable and publishable. Stencila's "Converters" module is a Pandoc-based collection of import and export routines. eLife's "Reproducable Document Stack" initiative is based on Stencila.

Basic Info:

Institutional host: independent
URL: https://stenci.la
Principal investigator: Nokome Bentley
Contact: hello@stenci.la
Lead developer: Nokome Bentley
Funding sources: Code for Science and Society, Sloan, eLife
Development partners:
Partners: eLife
Initial release: 2014
Version (as of June 2019): 0.28

Github (as of April 2019):

URL: https://github.com/stencila/
Language: JavaScript
License: Apache 2.0
Last commit: 2019-04-24
Contributors: 10

Tectonic

Tectonic is a modern LaTeX typesetting application, designed to be self-contained and easy to install. It automatically downloads support files so users don't have to install a full LaTeX system in order to start using Tectonic. Tectonic can use modern OpenType fonts and is fully Unicode-enabled.

Basic Info:

Institutional host: independent
URL: https://tectonic-typesetting.github.io/en-US/
Principal investigator: Peter Williams
Contact: peter@newton.cx
Lead developer: Peter Williams
Funding sources:
Development partners:
Partners:
Initial release: 2017
Version (as of June 2019): 0.1.11

Github (as of April 2019):

URL: https://github.com/tectonic-typesetting/tectonic/
Language: C, Rust
License: MIT
Last commit: 2019-04-22
Contributors: 17

Texture

Texture is an XML-based authoring and editing tool developed by the Substance Consortium, which includes PKP and eLife. Texture is a visual editor that natively produces a subset of JATS XML (inspired by JATS4R), which it encapsulates along with media and dependencies in its DAR file format. Texture offers a user-friendly editing XML interface, and can be integrated into other tools, such as OJS. eLife's Libero Producer is based on Texture, as is Stencila.

Basic Info:

Institutional host: Substance Consortium
URL: http://substance.io/texture/
Principal investigator: Michael Aufreiter; Oliver Buchtala
Contact: axfelix@gmail.com
Lead developer: Michael Aufreiter; Oliver Buchtala
Funding sources: partners
Development partners: eLife, PKP
Partners: eLife, PKP, Érudit, SciELO, EMBO SourceData
Initial release: 2011
Version (as of June 2019): 2.3

Github (as of April 2019):

URL: https://github.com/substance/texture
Language: JavaScript
License: MIT
Last commit: 2019-04-12
Contributors: 10

Vega

Vega is a media-rich authoring and editorial development platform hosted at Wayne State University Libraries. It offers a range of features and workflows to create, review, and share data, media, and text. Its ability to include information in a variety of representations (text, image, sound) makes it easier to communicate scholarly information to different audiences. Vega also supports typical academic publishing processes and gives users control over editorial and peer review workflows.

Basic Info:

Institutional host: Wayne State
URL: http://vegapublish.com
Principal investigator: Cheryl Ball, Andrew Morrison
Contact: publish.vega@gmail.com; lib.publishing@wayne.edu
Funding sources: Mellon
Development partners: Bengler
Partners: West Virginia University Library, Bengler, Arkitektur -og Designhøgskolen I Oslo
Initial release: 2019
Version (as of June 2019): 0.3

Github (as of April 2019):

URL: https://github.com/VegaPublish/vega
Language: JavaScript
License: MIT
Last commit: 2019-03-22
Contributors: 2

Vivliostyle

Vivliostyle is a CSS- and browser-based typesetting tool for digital and print publishing that adds book typography and layout capability of web browsers, supporting paginated EPUB and web publications or export to PDF. Vivliostyle complies with W3C standardization of CSS typesetting specifications. Vivliostyle.js was designed based on Peter Sorotokin's EPUB Adaptive Layout implementation.

Basic Info:

Institutional host: Vivliostyle Foundation
URL: https://vivliostyle.org/
Principal investigator: Shinyu Murakami
Contact: murakami@vivliostyle.org
Lead developer: Shinyu Murakami, Toru Kawakubo
Funding sources:
Development partners:
Partners:
Initial release: 2015
Version (as of June 2019): 2019.1.106

Github (as of April 2019):

URL: https://github.com/vivliostyle/vivliostyle.js
Language: JavaScript
License: AGPL
Last commit: 2019-04-22
Contributors: 12

Wax

Wax is a web-based word processor developed by Coko. It is the styling/formatting interface in use within Editoria, and the manuscript annotation and presentation portal in use in PubSweet platforms such as eLife's Libero Reviewer, and Hindawi's Phenom. Editoria provides context-sensitive tagging and formatting and a track-changes workflow, as well as many features driven by the needs of university press workflows. The initial version of Wax was based on the Substance.io library (as with Texture); Wax 2 is based on the ProseMirror library.

Basic Info:

Institutional host: CoKo
URL: https://coko.foundation/category/wax-editor/
Principal investigator: Adam Hyde
Contact: team@coko.foundation
Lead developer: Christos Kokosias
Funding sources: Mellon, Shuttleworth
Development partners: ProseMirror, Substance

Partners: Hidawi, Editoria
Initial release: 2018
Version (as of June 2019): active

Gitlab (as of April 2019):

URL: https://gitlab.coko.foundation/wax/wax-prosemirror
Language: JavaScript
License: MIT
Last commit: 2019-04-23
Contributors: 10

XSweet

XSweet is Coko's XSLT-based conversion/ingest tool for converting Microsoft Word documents (.docx) into HTML and beyond. XSweet extracts the contents of MS Word documents from their underlying XML into HTML, imported into an application, or used as a tool to convert it into another format altogether.

Basic Info:

Institutional host: CoKo
URL: http://xsweet.coko.foundation/
Principal investigator: Adam Hyde
Contact: team@coko.foundation
Lead developer: Wendell Piez, Alex Theg
Funding sources: Mellon, Shuttleworth
Development partners:
Partners: Editoria
Initial release: 2018
Version (as of June 2019): active

Gitlab (as of April 2019):

URL: https://gitlab.coko.foundation/XSweet/XSweet
Language: XSLT
License: MIT
Last commit: 2019-04-18
Contributors: 7

Zotero

Zotero is a desktop and/or network-based reference-management software for scholars. It has the ability to organize, collect, and format references and bibliographies for MS Word, LibreOffice, Google Docs, and other text-editing software. It also supports an enormous number of citation styles, and also provides a well designed document (web and PDF) collection and note-taking facility. Zotero is a social network that facilitates group collaboration, sharing, and publishing of reference lists. Originally developed in 2006 at George Mason U, Zotero is now a robust desktop tool as well as a full-featured web application, used by over 5 million scholars.

Basic Info:

Institutional host: George Mason University
URL: https://www.zotero.org/
Principal investigator: Sean Takats
Contact: press@zotero.org
Lead developer: Dan Stillman
Funding sources: Mellon; IMLS; Sloan; and by individual and institutional storage scubscriptions overseen by the Corporation for Digital Scholarship
Development partners:
Partners:
Initial release: 2006
Version (as of June 2019): 5.0.66

Github (as of April 2019):

URL: https://github.com/zotero/zotero
Language: JavaScript
License: AGPL
Last commit: 2019-04-23
Contributors: 47

Acknowledgements

This has been a complex and rewarding project. I would like to thank Terry Ehling and her colleagues at MIT Press and the Knowledge Futures Group (Amy Brand, Travis Rich, Catherine Ahearn, Heather Staines, and Gabe Stein) for their support and enthusiasm. Many people helped substantially with the environmental scan portion of this research; thanks so much to Kevin Hawkins, Peter Suber, Katie Shamash, the Radical Open Access Collective, and Gary Price. Thanks to the excellent people at so many software projects who took the time to talk and answer all our questions, to Josh Greenberg, Don Waters, and Patricia Hswe for their perspectives, and especially to Alex Garnett, Alison McGonagle O'Connell, Adam Hyde, Juan Alperin, Paul Shannon, and Chris Kelty for being over-and-above available to my questioning. Thanks to John MacFarlane, the developer of Pandoc and the Gitit wiki engine, for providing the tools we used to compile and write this report. Finally, this research has really been a group effort; thanks so much to Erik, Leena, Carmen, Kim, Avvai, Melody, Emma, and Ellen for being part of it!