# Open Collaborative Writing: Investigation of the Fork-and-Pull Model

EI PA PA PE-THAN, LAURA DABBISH, and JAMES HERBSLEB, Carnegie Mellon University

Work of all kinds increasingly takes place via networked digital environments that support diverse modes of collaboration. Previous work has investigated how collaborative writing takes place in shared and open workspaces. This study investigates how the "fork-and-pull" model of distributed version management, now widely utilized in software development, supports coordination and public contribution in collaborative writing. We employ a case study approach to understand coordination around written artifacts in an open pull-based environment. Through interviews and archival analysis of two open text projects: a mathematics textbook on homotopy type theory (HoTT) and open source policies by 18F, a branch of federal agency, we reconstruct how text artifacts were created. We find that in both cases an intensive multi-channel communication among a small core preceded an explicit release action in the form of public communication to solicit broader contribution. Our analysis reveals a dichotomy between two modes of collaboration: a perfecting mode of crafting a public written document versus an evolution mode. In perfecting, authors used the fork-and-pull model to manage contributions refining the text to a more correct version. In evolution, authors used the fork-and-pull model to use a written artifact as a template and starting point, customizing it to their relevant scope. Based on our results, we consider how we might design systems and policies to support pull-based coordination around written artifacts and the forms it takes.

CCS Concepts: • **Human-centered computing → Empirical studies in collaborative and social computing**.

Additional Key Words and Phrases: open collaborative writing, fork-and-pull model, open source model, coordination, case study, mixed-methods

**ACM Reference Format:**
Ei Pa Pa Pe-Than, Laura Dabbish, and James Herbsleb. 2021. Open Collaborative Writing: Investigation of the Fork-and-Pull Model. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW1, Article 137 (April 2021), 33 pages. https://doi.org/10.1145/3449211

## 1 INTRODUCTION

Digital environments enable a wide variety of tools with affordances that support diverse modes of collaboration. In collaborative writing, there is a long tradition of shared editors, from ShrEdit [38] to Google docs [39, 48], which are generally used by relatively small groups who edit the same document simultaneously or sequentially. Other research has focused on wikis, which enable a particular style of collaboration among larger groups of editors [26]. A newer "fork-and-pull" model, popularized by software developers under the banner of "social coding" [10], represents a new mode of collaboration that has arguably revolutionized software development. The foundation of this new model is the capability of easily replicating a document and making changes in a technologically isolated copy (or "fork"), then easily packaging any set of changes into a "pull request" and sending

Authors' address: Ei Pa Pa Pe-Than, pethan.eipapa@gmail.com; Laura Dabbish, dabbish@cs.cmu.edu; James Herbsleb, jdh@cs.cmu.edu, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, Pennsylvania, 15213.

2573-0142/2021/4-ART137. https://doi.org/10.1145/3449211

them to a maintainer of the original document, requesting that they be "pulled" into the original. **The goal of this paper is to better understand how the fork-and-pull model, transplanted from software development, influences the key phenomena of coordination and crowd contribution in collaborative writing**.

In this paper, we report the results of a mixed-methods case study of collaborative writing with GitHub. Our first case is a mathematics textbook on homotopy type theory (HoTT), and our second case is a set of open source policies adopted by 18F, a branch of a federal agency that consults with government agencies to develop open source software. We use interviews and an analysis of GitHub and other internet archives to reconstruct how the written artifacts associated with each case were created, focusing on how the teams of authors coordinated and how this coordination evolved over time, including examining participant roles and the kinds of contributions each project received from the public. We define coordination here as managing dependencies among activities, drawing from Malone and Crowston's definition [30]. For collaborative writing this involves integrating contributions including content additions and modifications, presentation and formatting, editing and more from multiple authors [16, 37].

We find for the HoTT book case, coordination in early stages was based primarily on "social locking" (i.e., assigning exclusive editing privileges to a particular person for a particular chapter) and direct contribution to the central repository. Coordination was aided by the use of a collection of interactions across technologies, including frequent in-person interactions, local mailing list messages, and notifications in GitHub. Opening the project to the public generated many small, corrective contributions, as expected, but also new mathematical content, corrections to proofs, questions leading to changes and clarifications, and suggestions about changing the writing process, sometimes accompanied by tools.

In the case of 18F open source policies, the origin was a fork of a set of policies from another government organization, customization of the fork for the 18F context by directly adding them to the fork, and addition of practices to augment the policies. Opening the repository was intended primarily to make it accessible to potential client organizations to inform them how 18F worked, but ended up generating a number of suggestions, questions, and edits that impacted the content. After the policy was released to the public, we identified about 80 forks of the 18F policies by other agencies, private organizations, and individuals. A number of these forks continue to be separately edited and maintained, serving the needs of other organizations.

Considering collaboration across cases, we observed similarities in intensive highly interactive ideation and content creation among a small group, followed by an intentional release action where contributors publicly promoted the written artifact and invited contributions on social media. We describe a distinction between perfection versus customization as two modes of community contribution to collaborative writing documents in the fork-and-pull model. We conclude with thoughts about the utility of the fork-and-pull model for enabling effective coordination following a transition to openness, as well as the unique combination of independence and connection between workspaces that it affords.

## 2 RELATED WORK: COLLABORATIVE WRITING

In this section, we will review previous research on collaborative writing with an emphasis on changes in tools and platforms over the years as well as features of these tools that enabled coordination.

### 2.1 Shared Editing

*2.1.1 Synchronous and asynchronous editing.* Synchronous editing refers to the capability of a writing tool that enables two or more authors to edit the document in parallel as in GROVE [13] and

ShrEdit [38]. This synchronous co-editing feature is supported in today's widely used cloud-based tools such as Google Docs[42]. Asynchronous editing, on the other hand, was common in earlier shared editors as in Quilt [14] and PREP [36], only affording coordination in sequence.

Different stages of writing and different numbers of authors often require different features and result in the movement of writing activity across tools [1, 43]. For example, Boellstorff et al. [4] experimented with scientific writing at scale with a small group of four authors, starting their project in Microsoft Word, migrating later to Google Docs which worked well for a small group of four authors in [4]. Tomlinson et al. [43] however found that this tool was inefficient for more than 25 authors collaborating on academic papers.

#### 2.1.2 *Commenting.*

Commenting or annotating enables the communication among authors and reviewers about edits or document content by adding notes to specific parts of the text as in earlier shared editors such as Quilt [14], Microsoft Word and Google Docs.

Commenting is not just an alternative strategy of direct editing but also used to maintain relationships with co-authors [2, 25]. Birnholtz et al. found that commenting was more common in an asynchronous editing mode whereas chatting was chiefly used in synchronous or real-time collaboration [3]. Olson et al. [39] found that the students used the document to temporarily store materials needed for their writing task such as links and references, requirements of the project, and discussion notes, coordination instruction, and the document outline.

#### 2.1.3 *Tracking changes and activity logs.*

Shared editors use several different mechanisms to provide awareness and the visibility of work among collaborators. One such mechanism is the "activity log" used by Quilt [14] which keeps a record of all interactions taken by the co-authors within the document. Another mechanism is "shared feedback" used by ShrEdit [38] in which all editing and commenting activities are presented within a shared workspace, enabling co-authors to maintain awareness of the progress of their work, much like "track changes" in Microsoft Word, and "suggesting" mode in Google Docs.

In general, as the number of authors and edits grows, editors encounter difficulty in keeping track of and reviewing edits [4]. In shared editors, edits are made directly on the shared document and shown inline or stacked together outside the body of the text, making it harder to trace changes, quickly grasp the overview of what changes were made, and coordinate among reviewers [5].

#### 2.1.4 *Social roles and isolation of workspaces.*

Some early collaborative writing tools tackled the coordination problem by supporting social roles based on the collaborators' responsibilities [36]. In Quilt [14], three social roles were defined: co-author, commenter, and reader, and each role was associated with a set of actions. PREP [32] created roles implicitly by dividing the workspace into three columns: the document plan or outline, the document content, and comments on specific parts of the document. ShrEdit [38] provided little structure, allowing authors to define roles and practices on their own. Google Docs enables the authorship group to assign roles of edit, suggest, and view, which are used fluidly in writing and editing [39].

ShrEdit [38] provided co-authors with a private editor as well as a public workspace. In a study of collaborative writing in Google Docs [41], Strobl found that 80% of the text documents did not have edit history, meaning it was prepared outside of the collaborative tool. Wang et al. [48], confirmed that writers sought privacy while collaboratively writing with others to avoid judgement and distraction.

### 2.2 Wikis

Wikis offer the capability to co-edit a document through a web browser. Contributors can collaboratively create, edit, negotiate, and communicate about the topic of a document in a shared publicly visible space [27]. Wikipedia[1] is the largest and most sustained collaborative writing project on the internet, and is built on MediaWiki software[2].

A MediaWiki page comprises four main features: the “article” page holds the main content, the “talk” page holds discussions, the “edit” page allows authors to edit the main content, and the “view history” page holds edit/revision history. Authors modify the text of the document on the edit page through wiki markup or in the visual editor displayed in a browser. Once done, authors “publish changes,” to the article page, and this explicit action integrates changes to the main document content. A combination of user type and the protection mode (which can restrict editing rights) determines the way that wiki users edit the document. Each article and user has a dedicated talk page for asynchronous discussion posts organized by time stamp. Prior studies [25, 26] found that editors use talk pages to discuss policy and practices, build consensus, and resolve conflicts [26]. Wikipedia users form self-organized teams, e.g., Wikiprojects[3], for management and coordination of a set of articles [15, 18, 35]

The wiki model enables the development of open documents where editing is done in a temporary document created once contributors initiate the edit action and removed once they trigger the merge action. Reviews in the wiki model are separated from the original document in contrast to shared editors where the evolving text and edits are synchronously present in the same document, and fork-and-pull model where comments can be made on the specific location of the document where the changes were made.

## 2.3 The fork-and-pull model

The fork-and-pull model is now the de facto standard of collaboration in open source software (OSS) between those who wish to contribute code to a repository (contributors) [19] and those who have edit privileges to the repository (maintainers) [10, 20]. The distinctive feature of this model is that in order to contribute to a project, a contributor makes a personal copy (called a fork) of all the files in the target repository. Desired changes are made in this fork, which functions as an isolated workspace. If the contributor would like these changes to appear in the original repository (rather than just maintaining them in the private fork), the pull request (PR) mechanism bundles them together into a PR, which consists of lines added and deleted to one or more files, and notifies maintainers, who can pull (or merge) these changes into the repository if desired.

GitHub is the most widely used platform supporting the fork-and-pull model of collaborative software development. It is used extensively in both corporate software development and open source software with over 100 million repositories hosted on the platform[4]. GitHub incorporates many social features alongside the fork-and-pull development model, supporting issue submissions to a repository, comments on pull requests and issue submissions. Previous research on GitHub as used by software developers investigated how transparency – the ability to accurately see ongoing and completed work – improves collaboration at scale [10]. Maintainers use pull requests as the primary contribution mechanism in order to trigger reviews of code contributions, but also to solicit contributions from the larger community [20] and to discuss bug fixes and new features [45]. Pull requests are more likely to be accepted when they fit the style and trajectory of a project [20] and when the submitter has some prior interaction with the project [44]. While forks are primarily used as a temporary workspace leading to a pull request to the main repository, forks are sometimes maintained independently, sometimes for good reasons such as maintaining customized versions of the software, other times resulting in inefficiency [51].

[1]https://www.wikipedia.org/
[2]https://www.mediawiki.org/wiki/MediaWiki
[3]https://en.wikipedia.org/wiki/Wikipedia:WikiProject
[4]https://en.wikipedia.org/wiki/GitHub

Contributors often observe pull requests in order to stay aware of project activity and use them to create a visible portfolio of work for career development and reputation purposes [19, 31]. They often suffer, however, from poor responsiveness of the project maintainers [19], and find it difficult to discuss anything beyond low level details of the particular contribution, and they often make use of additional communication channels [19]. Stakeholders outside the project may weigh in attempting to influence the decision [45].

There is some previous research on non-code projects in GitHub, e.g., exploring GitHub as a collaborative educational tool [50] and a tool to bridge collaboration among government organizations [25], and from the perspective of non-technology class [32] but only two studies specific to collaborative writing with GitHub i.e., an exploratory study of seven collaborative writing projects [28] and a study of writing practices of a specific group [17]. Manubot is a git-based tool designed specifically for collaboratively writing academic manuscripts, which supports editing via git-based code hosting services with some additional features such as continuous integration [23, 40]. Our work lies in the domain of collaborative writing similar to these studies [17, 28] but we systematically selected two cases, analyzing coordination as it evolves over time, differences in use of the tool associated with different content types, the transition from closed to open contribution, and the nature of crowd contributions.

The fork-and-pull model differs substantially from both shared editing and wiki models, providing different mechanisms for isolation (forking) and connection (pulling) between collaborators' workspaces. We are not aware of any studies that observe the entire life cycle of collaboration around a document written in a fork-and-pull setting. Investigating the development timeline and coordination evolution of a document provides significant insights into how the pull-and-fork model is used to support collaborative writing, and how it helps to accommodate openness. Hence our first research question:

**RQ1**: How does collaborative writing unfold in a fork-and-pull based open collaboration environment?

It is also important to understand what kinds of contributions authors receive from the broader community in a pull-based open collaboration environment. Tool functionality that makes it comfortable and convenient to make and review changes to a document, determine which of those changes are desirable, and make them visible and available to integrators, may facilitate external contributions to documents, as they do to open source software. Hence our second research question:

**RQ2**: What kinds of contributions do writing projects receive in a fork-and-pull based open collaboration environment?

We investigated these research questions in a case study of two writing projects hosted on GitHub, an open collaborative environment utilizing the git version control system and fork-and-pull model.

## 3 METHODS

We adopted a case study approach to examine the process by which open documents are produced in the fork-and-pull model. We chose the case study approach because it is the most suitable for addressing the question of "how" in a relatively new area of research [49], and has been applied successfully in previous studies investigating coordination practices of software development in open settings [22, 33]. We chose to study two open collaborative writing projects where a substantial portion of the work was done on GitHub. This section describes case selection, our data sources, and data analysis procedure.

## 3.1 Case selection

In selecting cases, we first searched for research on GitHub non-code projects with Google Scholar, using search teams such as **GitHub, non-code, book, document, and text**. In the retrieved list of papers (e.g., [28, 29, 32]), we looked for mentions of GitHub projects, and manually retrieved their GitHub links. Using the same query terms, we searched Google and retrieved news articles from Wired[5], readwrite[6], and mud[7], and retrieved GitHub links of the projects mentioned in these articles. In addition, we sent a query for pointers to the professional mailing list of the association of internet researchers (AoIR) asking members to identify GitHub repositories used for things other than software. We received 10 replies with 15 links to non-code project repositories in GitHub. After we retrieved the GitHub projects, we also collected the keywords used by authors of the research papers and articles where we found the links. The keyword list included books, papers, course syllabi, CAD diagrams, graphic design, journalism, music, policy documents, and recipes.

Next, to retrieve more projects of each type or possibly detect more types, we manually searched GitHub using the keywords we identified. We also looked at "tags or labels" of our initial set of non-code projects to understand other terms that GitHub users might use to refer to non-code projects, but we did not find much difference between GitHub user-defined tags and the keywords identified by authors of the articles. We further inspected the projects to identify the content type and extent of collaboration (approximated by summing the number of contributors, commits, forks, pull requests, and stars). We collected a list of projects of each type, noting the URL, content type, and extent of collaboration within each in Google Sheets.

Finally, we settled on a repository as our case unit, since each represented a distinct artifact with its own history. Since our focus is on coordination, we classified each of the candidate projects according to two opposing spectrums of widely-used task dependency types [46] as either "pooled/independent" (where contributions are made independently and aggregated into a collection, or "pool") or "team," (where the interdependencies among tasks are intensive). In particular, we looked at "commit and pull" activities of each project to determine if the majority of changes were either adding new content (pooled or independent work) or editing and integrating the existing content (joint/group work). Given that our research focus is coordination, and since pooled/independent work (e.g., creating a collection of recipes or course syllabi) does not require substantial coordination, we narrowed down our search by focusing only on projects with team work.

Our case selection procedure followed Yin's [49] theoretical replication strategy, in which projects are selected that are as similar as possible in all respects except for a known difference. Such replications tend to support interpretation of observed differences in the cases, as potentially related to the known difference between them. We decided to select one book, since this sort of gathering and structuring of knowledge content is similar in spirit to Wikipedia, and would enable us to compare our observations with the substantial body of literature focused on the open collaboration on that platform. For a second case, we decided to focus on policy which, in contrast to capturing knowledge, focuses on guiding and constraining behavior. In this sense it is loosely analogous to computer code, which "guides" a computer's behavior, and requires the writers to determine and express desirable human behavior. In both cases, we selected relatively large projects (in the number of participants) to give us a large body of coordination to examine. Both projects are similar with respect to the way that work was produced (joint/group work) but different with respect to the type of content produced.

[5]https://www.wired.com/2013/09/github-for-anything/
[6]https://readwrite.com/2013/11/08/seven-ways-to-use-github-that-arent-coding/
[7]https://www.makeuseof.com/tag/just-coders-9-ways-use-github-creative-work/

The first case we selected was a mathematics textbook on homotopy type theory (HoTT)[8] The second case we selected was a U.S. federal agency 18F open source policy[9] which was adapted from the policy of the Consumer Financial Protection Bureau (CFPB).

## 3.2 Data sources

We collected data from two main sources: semi-structured interviews and archival project activity records, which are described in detail in the sections below.

*3.2.1 Interviews.* We chose to conduct interviews with "core" and "peripheral" contributors from each project because we expected that they might have different purposes and strategies for contributing to the project. Following the way that the core and peripheral developers are defined in open source software development [33, 34] and also used by [10, 45], we define core contributors as those who made the most changes to the original documents, and peripheral contributors as those who made the fewest attempts to contribute. For simplicity, we pulled out 5 contributors with the highest number of commits and 5 contributors with the lowest number of commits from the project contributor page, and recruited them through emails and asking referrals from participants. Table 1 summarizes the overview of our interview participants.

We tailored our interview guides by contributor types: core and peripheral contributors. In interviews with core contributors, we focused on the project origin, involvement, the most recent experiences with the project, coordinating changes, in-person meetings, and perceived impacts of the project. In interviews with peripheral contributors, we focused on their involvement, motivations, and their contributions to the project. During the interview, we asked the participants to share their screens, and show us their most recent project activities, and probed for details about specific events they mentioned. The interviews lasted between 40 and 85 minutes and took place over Skype or Google Hangouts except one which was conducted over telephone. All interviews were audio-recorded, and all except two were screen-recorded. All interviews were transcribed and transcript documents were imported to Dedoose – an online collaborative tool for data analysis.

*3.2.2 Archives.* We used the following archival sources which formed a record of project activities: blog posts, wiki pages, news articles, public communication logs, and project GitHub activities. We manually extracted blog posts, wiki pages, and news articles and used Python scripts to collect public communication logs and project GitHub activities (refer to Table 2).

For HoTT book, we scraped the project wiki pages[11], and related postings on Carnegie Mellon University (CMU) website, Institute for Advanced Study (IAS) website, n-Category Café math blog, personal blogs, and online news channels including Aperiodical, Wired, Reddit, and Hacker news. We scraped messages from IAS group mailing list[12] which were indexed by message-id and date, and contained many different sorts of information including technical discussion, norms and practices, math discussion, and activity organization. For 18F policy, we scraped related postings on the agency blogs. For both projects, we scraped their entire GitHub activities including commits, pull requests, issues, comments of pull requests and issues, and contributors through GitHub API[13]. All commits and pull requests were indexed by sha.

[8]https://github.com/HoTT/book
[9]https://github.com/18F/open-source-policy
[11]https://ncatlab.org/ufias2012/published/HomePage
[12]https://groups.google.com/forum/#\protect\leavevmode@ifvmode\kern-.1667em\relaxforum/univalent-foundations
[13]https://developer.github.com/v4/

Table 1. Overview of interview participants

| Case | ID | Affiliation | Role in the project |
|---|---|---|---|
| **HoTT book** | C01 | Professor of Mathematics, Carnegie Mellon University | Co-organizer of Univalent Foundations program, manager, and maintainer |
| | C02 | Professor of Mathematics, University of San Diego | Developer of HoTT mathematical notations in TeX, owner of multiple book chapters, and maintainer |
| | C03 | Professor of Mathematics, University of Ljubljana | Technical director, digital editor, owner of multiple book chapters, and maintainer |
| | P01 | Software developer, telehash[10] on GitHub | Contributed two minor changes |
| **18F policy** | C04 | Senior advisor, U.S. General Services Administration | Founder of the 18F policy and practices |
| | C05 | Software developer, U.S. General Services Administration | Member of the 18F team |
| | C06 | Product manager, U.S. Digital Service | Member of the 18F team |
| | P02 | Product owner and front-end engineer, U.S. General Services Administration | Member of GSA involved in federal source code policy constructed from other open source policies including 18F's |
| | P03 | Front-end web developer, Bureau of Consumer Financial Protection | A member of CFPB responsible for opening the agency policy on GitHub that 18F adapted |
| | P04 | Owner, Open Tech Strategies LLC | OSS expert invited for commentary by 18F |

Table 2. Overview of archival data

| Case | GitHub activity | Posting from other archival sources |
|---|---|---|
| HoTT book | 3652 commits; 53 open issues; 524 closed issues; 10 open pull requests; 459 closed pull requests; 180 comments of pull requests; 4810 comments of issues; 84 contributors; 279 forks (8 organizations and 271 individuals) | 18 wiki pages; 9 blog posts; 8 news articles; 1 video; 1075 messages on group mailing list |
| 18F policy | 212 commits; 10 open issues; 27 closed issues; 0 open pull requests; 57 closed pull requests; 44 comments of pull requests; 277 comments of issues; 27 contributors; 80 forks (11 organizations and 69 individuals) | 6 blog posts |

## 3.3 Data analysis

Our dataset for analysis contained 10 interviews with core and peripheral contributors, and extracted case archives from both projects. The project GitHub activities were loaded into Google Sheets, and interviews and data from other archival sources were loaded into Dedoose for analysis. We used multiple data sources to perform cross-validation [22], and our analysis process consists of three phases but they were not necessarily done in sequence.

In phase 1, we analyzed our interview dataset following open and axial coding procedures described by Corbin and Strauss [8], and using three sensitizing concepts [6]: **coordination activity timeline, release process, and contributions**. We wrote case reports based on the results of our case-level analyses. In analysis, we started with open coding, individually analyzing the interview data, and creating descriptive labels or codes for text blocks. We also wrote memos as we analyzed the data. We generated 71 open open codes including in-person meetings, chapter owners, technical editor, direct push, social locking, creating awareness of current state, broadcasting changes, opening the project, a switch to push to pull, communication happened across multiple channels, external contributions, policy clarification, GitHub becoming a part of everyday life, publicizing through social media, and change in activity over time. In axial coding, we constructed a code hierarchy in which we grouped codes to represent higher-order categories or themes. We moved codes between groups until we detected a path that helped us present a theory of the production processes of digital artifacts in an open fork-and-pull environment. As an example of coded data, the issue conversation shown in Appendix A: Figure 4 describes the decision within our first case, the HoTT book to begin using pull-requests to review contributions. This event was associated with the open code "A switch from direct push to pull," nested within the axial code "From push to pull." Table 3 summarizes the axial codes generated from our analysis and presents example open codes associated with each axial code. Tables 6 and 7 in Appendix B represent the categories of pull-based contribution types we observed in each of our cases respectively.

Table 3. Summary of axial codes

| Stage | Axial Codes | Example Open Codes |
|---|---|---|
| **Pre-release coordination** | Original project | Project not adapted from any sources |
| | Forked | Project adapted from an original key project |
| | Ideation | In-person small group ideation and project definition, Ideate and create drafts through interactive in-person meetings |
| | Task distribution | Divide the whole artifact into multiple independent chapters, Assign owners to each chapter, Chapter level review, Settle on tools for collaborative writing |
| | Mutual coordination | Intensive content production, direct push plus social locking, Default to “push” mode, Assigning access privileges to a particular person |
| | Pull requests | Refining the artifact with pull requests, Vetting changes made by team members and experts before adding |
| | Multi-channel communication | Use various tools to coordinate work, Group forums, Communication with clients by email |
| **Release coordination** | From push to pull | A switch from direct push to pull, Change the access privilege of the majority of project team members, Vet incoming changes before merging with the released artifact |
| | Role evolution | Transform chapter owners to reviewers, Maintainers reviewing incoming contributions, Commenting on incoming contributions |
| | Social release | A call for a social action, Publicize the “done” artifact in online platforms |
| **Post-release coordination** | Extended community | Mentor the crowd, Helping contributors improve the quality of their changes to get merged |
| | Tree of forks | Fork and adapt each other artifacts, Modifying to the organization’s scope |
| | Continuous updates | Updating the artifact, Update with changes of external contributors, open vetting by a handful of project members |
| | External contributions | Changes made to the artifacts, Process changes, Math errors, Minor corrections |

In phase 2, we first constructed a GitHub activity timeline for each project, and thoroughly looked into bursts (or peaks of activity) in the timeline. This is because each burst represents the periods where most of the work occurred, and different bursts may represent different kinds of focused activities which may be important for the artifact production [7]. We then conducted content analysis of GitHub activities for each burst where we evaluated every entry of each activity (e.g. commits, PRs, etc.) and categorized them into types. Example types we developed include process change suggestions, questions that drove changes, custom content, and minor corrections such as typos and substantive content-specific corrections such as new math exercises and math corrections.

In phase 3, we inspected archived blog posts and wiki pages to have a better understanding of the goals and objectives of the projects and who was involved. As our analysis continued, we moved back and forth among these three phases to triangulate findings from other data sources. We aimed to improve the validity of our findings through this cross-validation of multiple data sources.

## 4 RESULTS

### 4.1 Case analysis: HoTT book

The HoTT book[14] of around 500 pages was written collaboratively using a combination of TeX and GitHub. The project originated in a year long program on Univalent Foundations (UF - a sub area of mathematics and logic computation), hosted by the mathematics school within the Institute for Advanced Study (IAS) at Princeton University in 2012/2013. The program gathered 98- academics in math and computer science from different institutions in residence for 1-2 semesters to work intensively and collaboratively on topics in UF. A subgroup of 7 theorists ran weekly meetings on 'informal type theory,' a sub area of logics and mathematics. After a number of meetings and activities, they decided to focus on writing a book on the topic of homotopy type theory (HoTT), a new area of logic and computation.

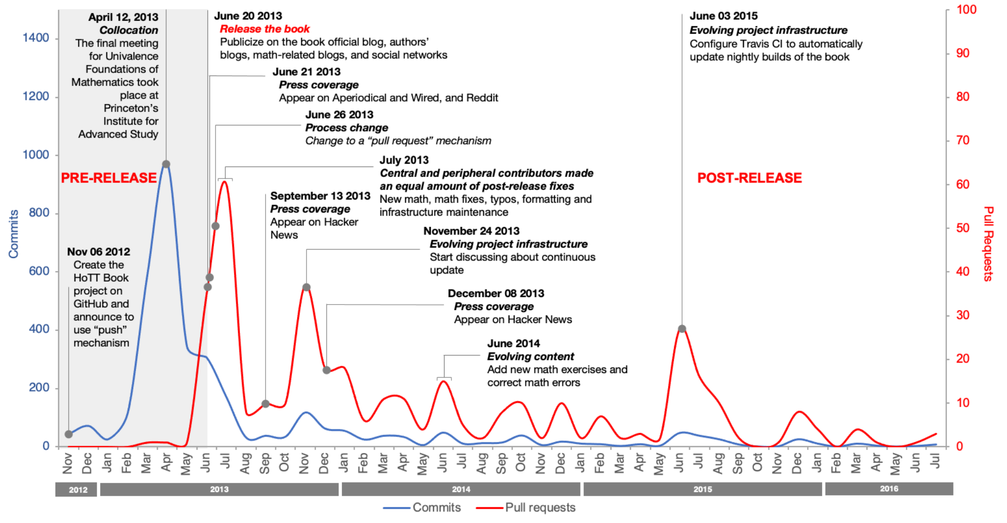


Fig. 1. GitHub activity timeline of the HoTT book. Commits refers to the number of direct changes to one or more files in the repository by members with write permissions. Pull requests refers to the number of indirect contributions made to the repository where a developer asked for changes committed to an external repository to be considered for inclusion in the project's main repository.

[14]https://github.com/HoTT/book

*4.1.1 Pre-release: Intensive in-person interaction and collaboration.* The small working group of 7 type theorists created an initial draft of the book during their residence at the institute (between September, 2012 and mid-April, 2013, see Figure 1). At the retreat, they posted an outline of the book on the retreat wiki which they used as a place to coordinate work and take notes, indicating who owned which chapter and the status of various review activities. Working group members wrote the book in parallel during the retreat, oscillating between intensive in-person discussion and independent writing. They used GitHub to post and exchange chapter drafts, and production started on Nov 6th, 2012 with the creation of the project GitHub repository. When chapter drafts were ready, they exchanged them for review, pulling them together into a manuscript.

After the UF session ended in mid-April the collaborators continued working remotely, refining the manuscript and chapters on GitHub. They discussed and coordinated revision through the Google Group mailing list until they were prepared for self-published release on June 20, 2013.

*4.1.2 Releasing the book: Growing the contribution network.* On the release date, the team posted messages announcing the book's release on the project's official blog, a math blog, and the members' personal blogs and social media accounts indicating that the project was done. They self-published the electronic and print book, transitioning from push to pull, resulting in growth of contributor network from initial 7 contributors in 2012 to 87 contributors at the time of writing.

*4.1.3 Coordination evolution: Openness forced a shift from push to pull.* Collaboration on the HoTT book evolved from intensive content production among a trusted core to perfecting by a large network of collaborators. The act of publicizing the book led to an influx of contributions from a wider set of previously unknown individuals. In the face of this, the coordination process evolved from production among a core to a gated review process where a subset of the original authors vetted and selectively accepted suggestions from the large network of individuals reading and using the book.

*4.1.4 Core members trust and high touch facilitated push access.* The pull request mechanism was seen as too cumbersome for the initial phase of writing, where a small group of chapter owners directly committed their changes to the GitHub repository as they made them. During the early stage of content production, the small set of core participants instead utilized a suite of informal coordination methods including division of labor and roles. The group discussed decisions through frequent interaction in person, via the group mailing list, and through notifications in GitHub. Once the group converged on a structure and outline for the book, they assigned specific people to be chapter authors (documented and updated on the wiki). Work assignments moved around fluidly for chapters, parts of chapters, reviewing and editing, as chapter drafts were exchanged through a 'social locking' mechanism (where authors by convention had exclusive edit rights to a chapter, then informed others in the group their section was ready for review). Lightweight informal roles helped the group avoid becoming bogged down in discussions about formatting and presentation, with one individual designated as the technical dictator with the final say on presentation and layout issues. This period of writing was characterized by high levels of activity and interaction as described by a core member of the author team:

> *"In the beginning it took some convincing and getting used to, although it was not too bad. In the end the repository served not only as an archive for our files, but also as a central hub for planning and discussions. For several months I checked github more often than email and Facebook. Github was my Facebook (without the cute kittens)." From Andrej Bauer's blog post[15] on June 20, 2013*

[15]http://math.andrej.com/2013/06/20/the-hott-book/

*4.1.5 Pull to vet external contributions.* The growth of new contributors after 'release' forced a shift from push, where all contributors could directly post their work and edits to the repository, to pull for quality control and review (see Appendix A: Figure 4). The original authors of the book were no longer creating basic content but mostly vetting contributions using pull requests in GitHub, dropping other tools used pre-release for coordination. As C01 described:

> *"So, before the book was released, we had a large group of people who had Commit access, and they didn't have to have pull requests; they just pushed everything in. And after the book was released, we shut that down, and there's a much smaller number of people who can pull. And now, I think it's just three or four of us who kind of manage the contributions."(C01)*

There was a dramatic change, then, in the nature of collaboration post-release, as the use of the other tools dropped to almost nothing, and issues (rather than the mailing list) were used to discuss contributions. Contributors changed from a small group with explicit assignments to large, unknown groups contributing anything, unpredictably. In this 'post-release' phase, the project received many diverse contributions from the now expanded network of contributors focused on perfecting the artifact. The early roles of technical dictator, technical editor, and chapter owner also seemed to disappear after release.

The shift from push to pull did two things. First, the pull request mechanism facilitated quality control, which allowed much larger numbers of people to contribute. Interestingly, it takes a lot of resources (maintainer time) to vet the contributions. The review demand associated with the pull model would be hard to manage if time was still taken up with writing original content. Also, presumably large numbers of contributors can only be accommodated when each contribution is small, on average. Second, the move to a pull model gave each would-be contributor the opportunity to work outside maintainer scrutiny until they felt their contribution was ready. Submission was a separate act – editing by itself (to a fork) was not submission for maintainer scrutiny.

*4.1.6 Opening as a social action.* In HoTT, there was a punctuated release moment where the book truly became 'open' for contribution by a wider audience. Although the book was open to public viewing prior to this action, contribution activity was limited to the small group of collocated authors at the IAS retreat. In order to release the book, the authors promoted the book publication and repository through blog and mailing list postings. Because the audience for HoTT represented both computer scientists and mathematicians who preferred physical books or needed a more traditional way to incorporate the book into their courses, release also involved making the book available on digital book stores and for order at the same moment/in a synchronized way.

Release, then, was a social/digital event, not a physical one. Releasing a paper book means placing it in stores, printing, shipping, etc. In an open collaboration environment such as GitHub, release instead meant announcing, "hey guys, here it is!". This was evident in the blog posts and articles made about the release by the core contributors:

> *"So we are inviting everyone to help us improve the book by participating on github. You can leave comments, point out errors, or even better, make corrections yourself! We are not going to worry who you are, how much you are contributing, and who shall take credit. The only thing that matters is whether your contributions are any good." From Andrej Bauer's blog post on June 20, 2013*

The book was always open to public viewing and editing in terms of permissions in the digital tool, but until it was publicized, it didn't attract contributions, since the public did not know it existed and was unlikely to stumble upon it.

*4.1.7 External contribution types.* Our analysis of pull requests to HoTT revealed that external contributions to HoTT included both modifications to content and process suggestions (see Appendix B: Table 6 for a summary). Post-release, the HoTT project received a wide variety of small fixes such as typographical corrections (similar to those observed for example in Wikipedia) but also received substantive corrections to proofs as well as substantial new content in the form of exercises, etc.

*4.1.8 Open environment facilitated process contribution.* HoTT received contributions suggesting new processes or infrastructure by use for the project. For example, in this issue (Issue #1221 - "Travis is failing"[16]), a core contributor, Mike Shulman[17] points out that their continuous integration system Travis CI is failing. A newer member of the community, Alitzer[18], not one of the original 7 authors and core contributors, makes a suggestion to adopt GitHub actions, which would change their project infrastructure:

> *"It might be a good time to bring up that Travis isn't the only CI we have to use. Recently we have Github actions which seems to satisfy all our needs. I wouldn't mind working out how to set it up. See: https://knapsackpro.com/ci_comparisons/github-actions/vs/travis-ci for a comparison with travis."*

This suggestion was ultimately adopted as the newer contributor worked to replace Travis CI with GitHub actions in a later issue and pull request (Issue #1247 - "Switch from Travis to GitHub actions"; PR #1227 - "Github actions").

The process and infrastructure contributions to HoTT book represent a novel and potentially unexpected positive side effect of open collaborative writing. While minor corrections and edits are expected given observations in previous settings like Wikipedia, such process contributions can only happen in an open, extensible, digital environment.

Because HoTT was produced in an open environment which was digital and extensible, process and tools could also be contributed by the crowd. Process contributions are a product of the project being hosted on GitHub, meaning it attracted particular kinds of people to use it (e.g., software developers). As one of the original contributors described, mathematicians in computer science departments and computer scientists focused on theory were familiar with processes used in a different domain of software development as part of the software focused aspects of their work and brought some of that to their work on HoTT.

*4.1.9 No errata necessary: Realtime error fixes and proof corrections.* In order to facilitate contribution, the book had to be far enough along for people to use but incomplete enough for the crowd to contribute. The minor and more substantive content contributions exemplified this. Typographical errors could be easily pointed out in the existing text and the identification of proof errors relied on the existing proof presentation in the book as it was. More substantive tutorials and exercises relied on the structure the authors developed and put forward where code-based exercises were attached to each chapter. Contributors could extend on this directly while attaching their submissions to the core structure of the book in a direct and easily indexed manner.

In contrast with traditional publishing, the online environment and immediacy of GitHub publishing meant that errors were fixed as soon as they were identified. Since the artifact was being used in classes and people's work, readers continued discovering errors, and adding content beneficial for others using the book as well. The continuous integration process meant authors were able to fix and correct the proofs and quickly integrate updates into the digital copy of the

[16]https://github.com/HoTT/HoTT/issues/1221
[17]https://github.com/mikeshulman
[18]https://github.com/Alizter

book available for download. This immediacy was in constrast to traditional mathematical textbook publishing process where authors compile an errata of such fixes and release it along with an updated version of the book over a period of many months to a year.

## 4.2 Case analysis: 18F open source policy

The second case we examined was an open source policy document created on GitHub by the digital consultancy “18F” within the U.S. government. 18F was seen as a “digital innovator” subscribing to the free and open source software (FOSS) development model, and implemented its digital tools and services through the reuse and sharing of source code. 18F also implemented an “18F incubation program” to bring new talent to public service. Figure 2 shows the production timeline and evolution of the 18F open source policy within GitHub.

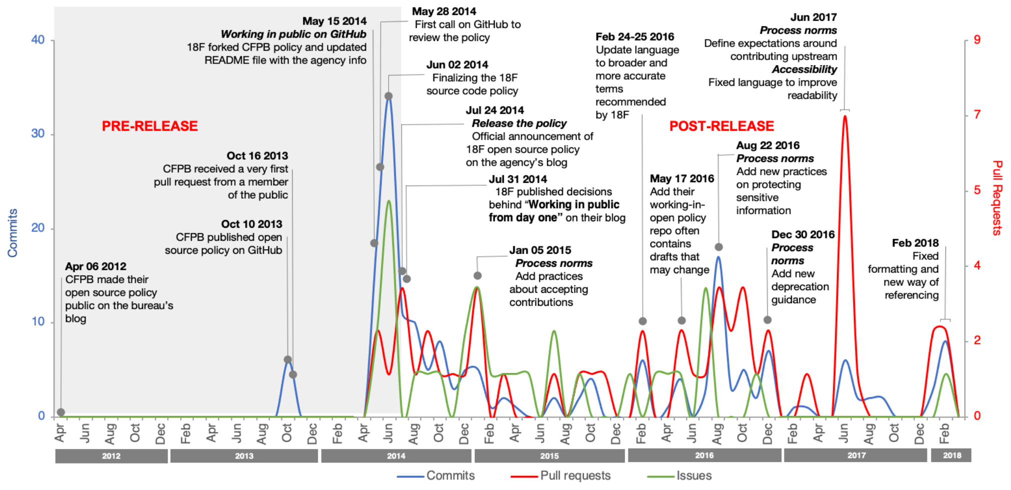


Fig. 2. GitHub activity timeline of 18F open source policy. Commits refers to the number of direct changes to one or more files in the repository by members with write permissions. Pull requests refers to the number of indirect contributions made to the repository where a developer asked for changes committed to an external repository to be considered for inclusion in the project’s main repository. Issues refers to the number of submissions received via the issues section within a repository typically composed of bug reports or feature requests.

### 4.2.1 *Forking and adapting CFPB policy.*

On May 15, 2014, 18F started its open source policy project by forking another key open source policy by CFPB which was refined through open conversations with the team members and expert review. The team had a total of 10 unique contributors make 52 edits/commits to customize CFPB policy for 18F. The initial edits included updating the policy with 18F information, clarifying language, fixing typos and formatting, setting contributor norms and licensing their policy under a creative commons license (CC0), attributing their work to public domain within the U.S., and other substantial edits. About two weeks after 18F started their policy making, the team started preparing to open their policy to the public.

### 4.2.2 *Announcement.*

On June 29, 2014, after three weeks of internal review and modifications, 18F posted a blog post entitled “18F an Open Source Team”. This post functioned as the announcement and publication of their policy. The blog post linked to their policy and explained how they were dedicated to open sourcing the code they created for their government agency clients. Three days after they released their initial policy, the 18F team encouraged others to open source their code through a blog post entitled “Working in public from day one,” describing their open working style and rationale for openness, including facilitating reuse and receiving improvements from other organizations that use their code.

*4.2.3 Promoting open source policy post-announcement.* After the announcements on their blog, the 18F team continued to promote their policy document by referring or sharing the link of their project repository on GitHub with potential client organizations. They also monitored relevant mailing lists to seek opportunities to spread the word. As C06 described:

> *“Also, you know, we’ll be on various mailing lists and if a question comes up that we’ve already figured out or we have an opinion on and that kind of thing, I can just say, ‘Oh, 18F does it this way. You can read more about it here.’ So it’s very useful for that reference so you don’t have to explain things over and over.” (C06)*

18F’s open source policy project evolved since their announcement, and as of this writing, there were 212 edits made by 27 contributors in which 155 edits were made by core contributors and 57 edits were contributed by peripheral contributors. This project currently has no pull requests/suggested edits to review. The contributors have raised 37 issues, and 10 of them were still open for discussion. The 18F’s open source policy was further adopted by several other organizations in GitHub including government and business.

*4.2.4 Working in the open as signaling: Symbolic value of tool selection.* For 18F there was a symbolic value to using a certain tool in order to signal that they were a certain type of organization. The use of GitHub was itself intended to indicate their agency’s culture was progressive, open, collaborative, etc. In looking at an organization’s tool selection, then, we shouldn’t assume that using any publicly visible collaboration software reflects only the utility of the features. In this case choosing GitHub was in a sense a "fashion statement" or proclamation of beliefs. The 18F team members were very conscious of this, as P02 described:

> *”We are committing to using open source software and to writing our own software in an open source manner, sharing with the public and encouraging that kind of collaboration. And so the projects on GitHub simply serve as a way for us to – another venue for publishing that to make it easier for people in the open source community to find and use GitHub pages to host it on another website. That’s what the project is.” (P02)*

18F was, in particular, sending a signal about openness as a value to their potential clients in other governmental agencies. Their policy in GitHub served as an advertisement and caution about their process. As our participants described, the repository and associated documents encapsulated 18F’s style of working and what clients should know prior to engaging them. Paramount among these considerations was the idea of working in public. The use of GitHub itself positioned them as a service provider with a particular brand. As C06 described:

> *“Other teams, because we are sort of a model team for doing more progressive work in government, you know, other teams will say ‘How do you get away with that or how do you do that?’ I can just point them to this and say, ‘This is how we deal with it.’ So that comes up a lot.” (C06)*

Through their tool selection, 18F promoted a new style of work in government. Their open source policy was an invitation both to potential future clients but also other organizations looking to evolve their own governmental agencies.

*4.2.5 Fork as the starting place.* The model of forking associated with the pull-based version control means the starting point is not typically a blank file, but related artifacts one can build on. In the case of 18F, they were able to draw from a reference CFPB was recognized as a leader who had done a great job of writing their open source policy. The 18F team, realizing they had a similar need as CFPB, decided to build off of their work. As P04 described:

> *"So it got started from them realizing that they had the same need that CFPB had and then they just started from CFPB's work and made their own modifications to it." (P04)*

CFPB first published their policy on their internal blog on April 6, 2012 by the agency's deputy chief information officer. The CFPB's policy was then moved onto GitHub on October 10, 2013 in order to enhance visibility and collaboration as well as to show the agency's commitment to open source.

A major feature of the fork and pull-based model is that you can build on and assemble existing artifacts into an approximation of what you want. This ability to fork an existing project is now common in software development where you don't start with an empty file, but you find something that provides some of what you want. In software this shift to building on others' work rather than writing code from scratch represented a fundamental change in how software is written [24].

Using someone else's work as a starting place relies on finding it in the first place. 18F started from CFPB's policy in part because they thought it was the best, and most leading edge. They came to believe that from both offline and online sources, through former members of the CFPB who later worked at 18F. If GitHub becomes popular for policy, others may in the future need to rely on GitHub signals to figure out what is the "best" policy to start with, much like GitHub developers use a variety of signals to find the best library for some purpose [10].

*4.2.6 Bridging tools across multiple audiences.* After forking and before being forked, collaboration in 18F had many parallels to pre-release collaboration in the HoTT book. Changes were done locally by a small group using a variety of tools, there was a "release" intended to get publicity (blog post, etc.) and then "perfecting" by the crowd with a variety of contributions including process contributions. Unlike the HoTT book however, 18F kept using a variety of tools to communicate with collaborators, where HoTT did not after release.

In 18F the team bridged different sets of tools to stay connected with different audiences. They had to continue using email and other means of communication to maintain connection with client organizations who were not developing software. At the same time, their software development activities took place on GitHub, and they connected on mailing lists and in other places with developers in other similar organizations working with the government. As C04 described, having conversations about the policy on GitHub brought a wider array of people into the process of policy making, but many still just sent emails:

> *"Having it on GitHub, I mean, it's hard to prove but it certainly felt like it brought in a wider array of people and made it more approachable for this sort of technology class to participate in processes that are often like, take place just on the Federal Register and just by lobbying shops who know how to, who know where all this boring stuff happens. It brought it to a different audience. And everybody who just doesn't want to use GitHub and just wants to send an email, they can still do that. And believe me, they do. But for folks who might not be used to interacting with the government in that way could be in there in GitHub, gets – generates more participation and humanizes the process to that class of folks." (C04)*

*4.2.7 Paying it forward: Other organizations forking and adopting.* Just as 18F adapted their policy from CFPB, many organizations built on 18F's policy to form their own, perhaps they conceived

18F as a "digital innovator." Other organizations looked to them as a model of how to develop technology and work in the open. The 18F contributors were aware of this use in part because the forks were visible but also because they interacted with these types of users occasionally in the pull requests or issues. As C06 described:

> *"So I think a lot of people externally see this policy as kind of the most forward thinking policy of its kind or the strongest policy of its kind, so other people sort of ask questions and hope that we're going to work out what our answer will be and then they'll use that. So it's kind of the gold standard." (C06)*

Policy fundamentally is a set of rules people follow, and other organizations can adopt and adapt existing policy to their own settings and circumstances. The 18F policy was forked 80 times on GitHub by 11 organizations and 89 users including U.S. federal and local agencies, e.g., U.S Customs and Border Protection forked the 18F policy project on July 6, 2016, and then updated it with the agency information, and the approval process of source code release (see Appendix A: Figure 5). Other entities that adapted 18F policy include National Park Services, NYC planning, USAJobs, local and international companies, and other anonymous entities. In adopting 18F's open source policy, contributors were mindful of scope adapting the policy to their own organization.

There is a notion of scope for policy documents, then, that references the organization, entity or set of people the policy applies to. The scope of the work is critical to how an organization might leverage and build on other's policy in an open environment. Policy has a scope limited to the organization that's adopting it. The edits and modifications made to a policy document adopted from a different organization are not relevant to the world (e.g., the HoTT book is for anyone who is interested), but relevant only to the adopting organization or entity, i.e., the one whose policy it is. So collaborative editing of a policy document is perfecting only within a particular scope.

*4.2.8 Nature of contributions.* The 18F organization used GitHub primarily as a tool to educate potential clients on the OSS process rather than for soliciting contributions. 18F's policy document received contributions as it was discovered by successive organizations. The contributions it received included questions that drove changes, suggestions for improving accessibility, and language. See Appendix B: Table 7 for a summary of the types of contributions 18F's policy document received.

**Questions drove changes**. The project evolved in response to use and exceptions that arose during use as 18F members worked with other government agencies. The 18F team members indicated, e.g., external collaborators would ask questions about their work, or how to apply their style of working, which generated the need for clarification. As one participant (C06) described:

> *"Changes to that repository often happen in response to questions or conversations or problems that come up in interacting externally . . . Okay, how do we deal with contributor license permits? . . . So that's an example of a policy changing in response to something external, even though it wasn't a direct contribution." (C06)*

In this way, outsiders asking questions about the policy made contributors aware of the uses and needs they had not yet considered. They evolved the policy in response to these needs as they learned about them.

**Improving accessibility**. A large proportion of changes were focused on making the project and associated communication documents more accessible and welcoming to a broader audience. This included pull requests asking for language changes to make the project more accessible, e.g., PR #19 - "Adding some friendly introduction text" which was subsequently refined by a core member[19] with PR #39 - "Integrate code of conduct into contributing document, tighten up intro a bit making

[19]https://github.com/konklone

language less intimidating" to condense the text and make the language friendlier ("especially since we're mentioning a set of rules now)".

**Extending policy**. Some of the wording changes resulted in extending the policy itself through more general and concise wording. This includes pull requests suggesting the use of general neutral pronouns or avoid pronouns entirely, e.g., PR #68, "I use 'they' as a pronoun, and noticed that this licence uses 'his or her' wording, so this pull request makes it easier for me to use the licence. It might also be worded *waiving all rights to the work* to avoid pronouns entirely" (see Appendix A: Figure 6).

**Clarifying project process**. An important function of 18F's repository was explaining to other organizations how the team functioned. As a result many of the changes and requests for change were associated with explaining or clarifying process norms (Project process info). This meant improving the contents of the project documents or posts related to them to provide information that was missing or improve existing information. A minor example of this was communicating a change in the appropriate hashtag to use (e.g., PR #62, "On practices page, updating #compliance to #infrastructure"). A more substantive modification was explaining that the work process was iterative (e.g., PR #52, "add information to make clear that our repos are iterative and what people see may not be a final version").

Another type of process clarification related to aspects of how work is done in OSS more broadly. For example, through PR #22 - "Add paragraph to team practices doc encouraging upstream contributions", 18F encouraged members to contribute back to projects that 18F depends on, and through PR #29 - "Add section to practices doc about accepting contributions", 18F declared that they welcome contributions from the public.

## 4.3 Case synthesis

The HoTT book and 18F open source policy collaborations represent two examples of writing projects carried out in an open collaborative pull-based environment. In the case of the HoTT book, the intention was to represent a particular branch of mathematical knowledge as accurately and clearly as possible, and opening allowed the crowd to contribute to this effort, employing its "long tail" of varied expertise. In contrast, the intention of the 18F open source policy was to create a document that would guide the behavior, in a particular domain, of one specific organization, 18F, and to represent these intentions to potential customers and other observers. Rather than helping to ensure accuracy and correctness, the crowd was invited to help ensure the policies expressed what the organization ought to do, and conveyed this in a way that was attractive to potential customers. Here, we consider the similarities and differences we observed across the two cases to understand the nature of writing in this environment. Our goal is to build up a general understanding of writing in an open collaborative environment and work towards theory about how properties of the written documents themselves influence the value of openness and nature of coordination in this setting.

*4.3.1 Rich early collaboration phase for ideation and project definition.* In both cases collaboration unfolded after an intensive phase of high interactivity, multi-channel coordination among a small set of contributors. In the HoTT book, this involved initial planning and document structuring at in person meetings, documented on an internal wiki, with the 7 individuals working most closely on the book, followed by frequent discussions, mailing list interactions and changes directly to the document itself (without use of the pull mechanism) using social locking as a primary coordination mechanism. After release, the core group shrunk to a handful, and their roles changed from producing content to vetting content for the larger crowd. In 18F, after forking the CFPB document, collaborators used issues within their repository along with in-person conversations and emails among team members and with external reviewers to refine their document in addition to directly committing changes to the repository. A rich early collaboration phase among a small core supported ideation and project definition. This initial work defined a shared goal and the team used whatever means possible to work towards that goal. The early phase ended at the point where the original core editors decided that the document was sufficiently complete to have some general utility beyond the core group. This appeared to mean that it likely had no glaring errors that would cause embarrassment, nor such a lack of clarity that it would confuse external readers.

*4.3.2 The release action: What it means to be open.* The work on both HoTT and 18F suggests that truly opening an online artifact actually implies an explicit invitation to view and contribute. Across both cases, there was a publicized release action that indicated that the written artifact was ready for consumption, even though it already was publicly viewable in terms of permissions and access control. Release was a call for social and digital action, a key event that acted as a catalyst for letting the public and potentially interested readers know that the book was ready for their attention or consumption. This was achieved in HoTT by “actively” publicizing and promoting their book on several platforms that the potential target audience group used (blogs, mailing lists, etc.), and in 18F by making several blog posts about their policy.

The social release action used in HoTT and 18F, and the community response suggests putting work in a public repository is not sufficient, it also requires a notification that something new is no longer changing in fundamental ways, and is ready for consumption. This was an important phase transition in both cases. This connects with the idea of ‘release‘ in software, where developers mark a moment in the constantly evolving code base as the point at which it is ready for public use. Interestingly, in both cases, the announcements were primarily about the artifact being ready to use, rather than primarily a call for contributions. Explicit release is in contrast with continuous editing in a Wiki, where there is no equivalent release action, and work is just a progression towards quality and maturity. Even “stub” articles, which have basically no useful content, are publicly available and as easily found as more mature and complete articles. In the case of HoTT, the public webpage announcing the book and blog posts created by its authors acted as a notification that something new was ready for consumption[20].

Since the release action triggered contributions from a much larger and less trusted group, coordinating and vetting these contributions is potentially more difficult. Anticipating this, the HoTT authors changed to a "pull request only" coordination model, meaning that each contribution would be vetted by a trusted editor. The 18F organization began by adopting the CFPB policy which was already hosted in GitHub, and did not really anticipate external contributions. Nevertheless, for 18F just as for HoTT, the pull request mechanism made contributions from a potentially large and unknown audience possible.

*4.3.3 Perfecting vs. Evolution.* The collaborations around the HoTT book and 18F open source policy document represent two distinct forms of writing through an open pull-based environment: perfecting vs. evolution. By “perfecting,” we mean expressing desired content as clearly and thoroughly as possible. By “Evolution,” we mean editing content to make it more suitable for a particular context. Here we consider how key differences across the cases describe these two modes of open collaborative writing in terms of 1) properties of the artifact itself and initial collaboration and 2) interactions with the broader audience in an open environment. Figure 3 presents a visual comparison of these two modes of collaboration. (See Appendix B: Table 8 for cross-case comparison of the distinction between perfection and evolution styles of open pull-based writing collaboration observed.)

[20]https://homotopytypetheory.org/book/

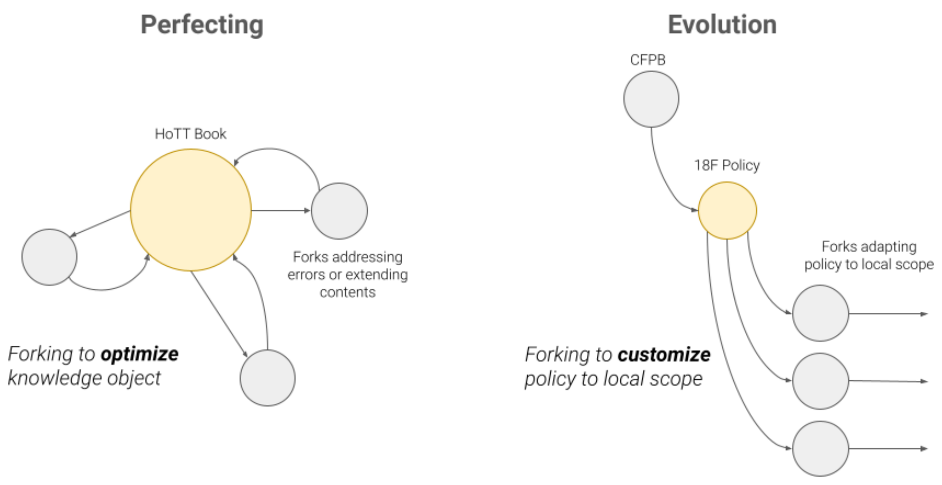

Perfecting
HoTT Book
Forks addressing errors or extending contents
Forking to **optimize** knowledge object
Evolution
CFPB
18F Policy
Forks adapting policy to local scope
Forking to **customize** policy to local scope

Fig. 3. Perfecting versus Evolution Modes of Open Collaborative Writing via Fork-and-Pull Model. Yellow circles represent the focal repositories in each case, gray circles with incoming arrows represent forks of the repository.

**Artifact: Ground truth vs. living document**. An inherent difference across our two cases is the nature of the artifact itself. In the HoTT project, participants were creating a book intended to encapsulate knowledge which is more like an article in Wikipedia. In contrast, 18F's open source policy document encapsulated a set of rules applied to a setting, in this case the software and government organizations 18F worked for. Thus there was a distinctly different goal in terms of what each project was creating from the get go.

As a function of the different artifact purposes, each artifact had a distinct life cycle. For HoTT, because the goal is describing an area of type theory in mathematics, there is an idea of the document converging to a ground truth version. There may be future editions that contain additional proofs, but they will effectively create another single document that takes the place of the original. Because the corresponding print and digital versions are intended to stay in sync with this single document, the publisher will issue errata and new editions, and constantly update the stable e-version to match the print one. It does not make sense to think of multiple versions existing in parallel. A new version implies the obsolescence of the old version.

In 18F by contrast, there is a continuous evolution of successive forking and adaptation of the policy document. 18F's open source policy is itself an iteration of the one created by CFPB. It has been adopted and customized by many other organizations, through the forking mechanism in git[21] and on GitHub to apply to the scope of the particular adopting organization. In this way there is a goal of optimizing the set of rules within the policy to suit the context of the particular organization rather than a final notion of ground truth (as in HoTT). The document life cycle is one of evolution, with an origin elsewhere and potentially many permanent versions afterwards. Because the document represents policies that were enacted, there is a need to synchronize the written document with the associated practices implementing the policies. The downstream and

[21]https://git-scm.com/

upstream adaptations continue to evolve with use by their new organizations, and so 18F members found utility in monitoring among the different versions to find useful changes.

**Openness: Optimizing vs. customizing**. Another inherent difference across our two cases, was the perceived or expected role of the broader audience in the open pull-based environment. For HoTT their intention and expectation of opening the artifact to the broader community, or crowd, was to help perfect the knowledge object they had introduced and optimize it to get closer to ground truth. GitHub in this case was purely a tool and mechanism for openness, chosen in part because the audience for their work were software savvy mathematicians often found in computer science departments. Choosing GitHub meant they could leverage people who were already there and familiar with the open pull-based way of working on software.

On the other hand for 18F, their goal in opening their policy to the world by posting it on GitHub was to signal their values to potential clients. They wanted to tell government agencies how they would work with them and set expectations about the code they were creating. Their intention was not necessarily to receive contributions from those agencies to refine and improve the policy – in fact, the primary blog post announcing the open source policy did not mention the possibility of a larger audience contributing to it. Moreover, here was not the same notion of ‘correctness’ or ground truth. GitHub in this case had a signaling value because of the philosophy and style of working, beyond simply being the means for openness and pull-based collaboration. Because their government agency clients were not necessarily on GitHub, they had to bridge multiple communication channels to increase awareness of the policy. This extra work, however, aligned with their goal of evangelizing the open source methodology and bringing people into the open mode of collaboration

## 5 DISCUSSION

Our analysis and observed collaboration patterns across cases suggest important differences from existing collaboration tools. First, a set of forks of a document form a graph which connects independent workspaces to the original document. Second, these forks can be durable over time affording customization of a document for specific purposes independent of the original documents continued evolution. Third, annotations are directly connected to the relevant content which supports conversation around a change but constrains discussion to incremental evolution of content. And fourth, the transition to opennness is marked by a social action via other platforms. Table 4 summarizes these factors and we consider them here in contrast with the properties of shared editors and Wikis.

### 5.1 A set of forks form a graph connecting independent workspaces

In our two-case study, we saw that the fork-and-pull model created a graph of connected independent workspaces in collaborative writing. Independence is created by the forking mechanism, which allows any collaborator to fork the global workspace, creating a local copy. This is somewhat similar to the “private workspaces” in some shared text editors like ShrEdit [38], and the common practices in Google Docs to prepare text privately before pasting into a shared document [48]. The fork-and-pull model is specifically designed to create independent workspaces and make movements of edits among them simpler.

### 5.2 Forks can be durable

Unlike the temporary workspaces in MediaWiki that survive only until changes have been pushed to the master repository, forks in the fork-and-pull model can be durable. They make it quite convenient to create a separate permanent repository, customized for different needs. This happens

Table 4. Key attributes of the Fork-and-pull model contrasted with shared editors and media Wikis.

| | **Shared editors** | **Media Wiki** | **Fork-and-Pull Model** |
|---|---|---|---|
| **Networked workspace** | Shared / completely overlapping | Temporary connected workspace while editing | Separate, isolated but networked workspaces |
| **Workspace durability** | No separate workspace persists post-revision | Workspace only lasts until edit is saved | Durable forks |
| **Discussion** | Via comments on text | Separate discussion space affords richer debate | Linked directly to content, must be augmented with other tools to provide discussion space |
| **Openness** | Among small group of editors with access | Public to the world | Public with social release indicating readiness for contribution |

in software development [51], and, in a similar way and for analogous reasons, it happened in our policy-oriented case of 18F.

Even though permanent forks maintain independence of workspaces, the forks are visible, so anyone can monitor other forks in their project network for potentially useful changes, and consider pulling them into one's own fork. This helps to maintain awareness of their connected workspace. One interview participant from the 18F case mentioned occasionally monitoring the originating policy for changes that might also be useful for 18F. While no such changes were reported, this sort of monitoring is common in software development [10], and we speculate that it may become more common among sets of related forks with customized versions of a common document. The pull request mechanism makes it simple, for example, when a mistake is corrected, or an improvement made in the original document, to pull this change into a customized version.

As shown in the 18F case, the fork-and-pull model can make document construction particularly efficient. Starting with a document that is a good approximation of what is needed is clearly an advantage, where legal and ethical considerations permit it. This copy can be independently maintained, and continue existence as a customized version. The bulk of the document may never be edited, as in the 18F case. In software development, the availability of useful pieces of software for general use has become so widespread that it is often referred to as our "digital infrastructure," which greatly enhances the speed and quality of software development[12].

## 5.3 Annotations directly connected to relevant content

All of the collaborative writing platforms we reviewed support some form of annotation in order to communicate plans, discuss next steps, moderate disagreements, etc. In some shared editors, like PREP [36], a space is specifically dedicated to annotations. In less structured editors like ShrEdit [38], users worked out systems for commenting, voting, and maintaining awareness within the document itself [11] in contrast to Wikis where commenting takes place in a space separated from where changes are made. In the fork-and-pull model, comments are attached directly to the part of the artifact to which the comment refers, e.g., pull requests, commits, or issues. Particularly in the case of 18F, issues and comments on them were used extensively both for project management and as a way of accepting and responding to comments from the larger community. This highly structured commenting functionality makes it very easy to provide comments on specific items, but it makes it difficult to make more strategic comments that concern the whole project, future directions, or other issues that are broader than a single item. We observed that these kinds of comments were made in other forums, such as Google Groups, blogs, or shared editors. The structure makes some kinds of comments very easy, and the referents clear, but limits the ability to comment on issues not tied to a particular item.

## 5.4 The transition to openness

Research in the literature on collaborative writing does not reflect the kind of phase change transition to openness we observed in both of these projects. While private editors as in ShrEdit [38] and transitory private editing in MediaWiki[22] reflect a private-to-public transition in the context of individual contributions, documents are typically open to the group (in shared editors) or the the world in MediaWiki. In open source software development, there is typically a "release" of the software, which amounts to a declaration that it is ready to use [22], even though repositories are typically visible to the public from the beginning. The transitions we observed in both HoTT and 18F more closely resembled a software release, by publicly declaring the document ready to use.

This transition point also marked an important shift in the collaborative process, taking advantage of the flexibility of the fork-and-pull model. Early on, small groups of trusted contributors directly pushed content into the repository, while after opening, the larger groups attracted to the open document submitted changes for review through the pull request mechanism. The flexibility to operate in both modes simplified opening while protecting early contributions from premature attention and distraction.

This kind of support for transitioning can also be seen as enabling effective coordination, similar to what [26] observed in Wikipedia. Small numbers of initial contributors who created the structure for a page set the stage for implicit coordination for subsequent editors. This seemed to allow larger numbers of editors to contribute effectively. This is similar to what we observed in both of our cases, which began with relatively small numbers of authors, and publicized to the crowd only after a useful version had been produced, providing structure for the crowd.

## 5.5 Implications for fork-and-pull based collaborative writing

Our case observations suggest a set of implications for leveraging the fork-and-pull based model to support collaborative writing activities. We summarize here each of our key insights from across our cases and consider the implication of each for supporting open collaborative writing via the fork-and-pull based model. Table 5 provides a summary of our key insights and corresponding implications.

We observed that forking created a network of independent but networked workspaces which allowed for both perfection and evolution of a written document. At the same time it was difficult to maintain an overview of the full range of the different forms of modifications being made to a written document. The network view provided in GitHub is difficult to traverse and provides only commit level indications of modification once they are made publicly visible in a contributors own repository. Better support for activity summarization and identification of duplicate work could support cross-fork awareness and overviews for document maintainers.

There is opportunity to provide better support for tracking relevant changes across forks of a document. The persistence of forks over time allowed organizations to customize 18F's policy document, which 18F themselves had adapted from the CFPB. The durability of forks afforded this

[22]https://www.mediawiki.org/wiki/MediaWiki

Table 5. Key insights and implications for fork-and-pull based collaborative writing support

| **Key insights** | **Improving pull-based tools (GitHub) for collaborative editing** |
|---|---|
| Forking mechanism created network of independent workspaces for collaborative writing | Support activity summarization and identification of duplicate work |
| Durable forks supported ongoing customization and adaptation | Simplify upstream and downstream monitoring |
| Annotations directly connected to relevant content, Does not have separate discussion space, use augmented with other tools to provide this | Provide project-level and synchronous discussion capability |
| Transition to openness supported by social release action | Provide mechanisms for conveying project state in terms of contribution readiness |

ongoing evolution with provenance and document history. At the same time, the team responsible for each child document had to do a great deal of manual checking to monitor upstream and downstream modifications to their policy over time in case they were relevant. Better support for upstream and downstream monitoring of changes could utilize intelligence to notify document owners of potentially relevant changes within the network of forks.

Discussion and annotation is another area of opportunity for better support. Without a separate discussion space, annotations and discussion of document issues remained fairly granular, tied to specific lines and sections of a document. For both of the projects we examined, core contributors had to supplement interactions on GitHub with other platforms such as mailing lists, email to have richer discussions about project level decisions. Collaborative writing platforms that support the fork and pull model need to consider how to complement the detailed transactional comments on pull requests and commits with other forms of discussion and social interaction. Longer form communication like email lists or more synchronous forms of communication like face-to-face interactions or messaging may augment pull-based tools to support higher level decision discussions but their record remains disconnected from the contents of the work itself.

Finally, our observations suggest the importance of the social 'release' action for open collaborative writing projects. We obseved that although both cases in our study created their projects in a public repository on GitHub, they required a social 'release' action to signify they were ready to accept commits. Collaborative writing platforms should consider providing mechanisms for conveying project state in terms of readiness for contribution by a broader audience.

Our work also has implications for coordination theory. It has long been recognized that dependencies among tasks create coordination problems of varying difficulty and complexity [47], and that the particular patterns of dependency generate different coordination problems requiring solutions, "social algorithms" that address them [21, 30]. The current work extends these ideas by noting that degrees of openness are associated with different patterns of dependencies that seem best addressed by different tools and modes of coordination. Early in the process, with small closed teams, social conventions and a divide-and-conquer approach allowed the rapid generation and review of large bodies of text. Later, as the attention of a larger community was drawn to relatively complete drafts in order to perfect the work or to tailor it to particular contexts, the pull-and-fork model allowed much more transparent environment and fine-grained coordination for accepting and reviewing smaller contributions from less trusted community members. The needs of the task, the size of the community, and the nature of contributions and perhaps other factors seem contingently related to the tool characteristics of work isolation, discussion spaces, and transparency, such that tool features must be carefully matched to the ongoing work.

## 6 LIMITATIONS AND FUTURE WORK

As with any research study, there are limitations to our research method and approach that may limit the generalizability of our observations. First, because we took a case study approach it is not possible for other researchers to exactly replicate our work. At the same time, most if not all of the artifacts and interactions we analyzed are publicly visible through project repositories for each of our cases mitigating this concern a great deal.

Another potential limitation is the number of interviews we were able to obtain on each project. Because we were only able to talk to a subset of contributors within each project there may be important details of collaborative activity not captured in our data collection. At the same time, we attempted to sample for breadth to obtain perspectives on collaborative activity from individuals in different roles on the project and with different levels of participation. In addition, we triangulated our interviews with archival project activity logs, documents, and interaction histories to provide a more comprehensive view of collaboration history and capture events that interviewees may not have mentioned. Member-checking would be another alternative method of triangulation to undertake in future work[9].

The generalizability of our results may also be limited by the case study approach and associated narrow sample of projects supporting depth versus breadth in the analysis. We focused on two projects, each one representing a particular type of written artifact, one textbook and one policy document, versus analyzing a broad range of different types of writing projects. This means our results may not generalize to other types of written artifacts, such as recipes for example. Future work should examine the extent to which types of written artifacts exhibit similar or different collaborative patterns in use of the push versus pull model and benefits from openness.

Finally, there is a potential that our results may not generalize beyond the two projects in our sample. Because we focused on two specific projects each representing one type of written artifact, it is possible the interactions on those projects were idiosyncratic for that artifact. Future work should examine additional projects working on the same or similar types of documents (textbooks and policy documents) to examine the extent to which the patterns we observed generalize to those types of artifacts. This work can also help identify important differences in the nature and variety of perfecting versus customizing patterns and boundary conditions in the way these behaviors are carried out.

## 7 CONCLUSION

Our case study of collaborative writing using a fork-and-pull model makes five contributions to the literature. First, we showed how coordination unfolds over time, from socially-managed coordination where a small group of writers pushes content directly into a repository, through opening up to the crowds, where contributions are managed by means of pull requests. Second, we showed how this process works differently in two domains: capturing and expressing knowledge and writing policies to govern aspects of organizational behavior. Third, we described how the crowd contributed to the writing process, in ways both small and significant. Fourth, we showed how the same technology mechanisms of forks and pull requests were used either to "perfect" a single document or to create different enduring versions, customizing to particular organizations. Finally, we contributed to coordination theory, showing how two stages – closed then opened – of open collaborative writing present very different coordination needs that seem to be best addressed by different styles of tools.

Future directions for research on the fork-and-pull model should explore the evolving model of documents not existing in isolation, but being embedded in a graph of forks and edits, as useful content from any document is incorporated into others, built upon, and potentially opened to a new crowd. One could imagine ecosystems of documents, say, inside an enterprise, much as software currently lives in ecosystems of interdependent programs [5]. This could potentially speed the creation of new documents (such as proposals, policies, bids, or descriptions) and improve quality as corrections and improvements can be considered on the common elements across the graph. The profound change that has transformed software development, enabled by openness and tools that can take advantage of it, may have echoes in the world of collaborative writing.

## ACKNOWLEDGMENTS

We thank our interview participants for sharing their experiences. We also thank the CSCW associate chairs and reviewers for their helpful feedback and guidance through the revision process.

## REFERENCES

[1] Andrew Begel, Jan Bosch, and Margaret-Anne Storey. 2013. Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder. *IEEE Software* 30, 1 (Jan. 2013), 52–66. https://doi.org/10.1109/MS.2013.13
[2] Jeremy Birnholtz and Steven Ibara. 2012. Tracking Changes in Collaborative Writing: Edits, Visibility and Group Maintenance. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work* (Seattle, Washington, USA) *(CSCW '12)*. ACM, New York, NY, USA, 809–818. https://doi.org/10.1145/2145204.2145325
[3] Jeremy Birnholtz, Stephanie Steinhardt, and Antonella Pavese. 2013. Write Here, Write Now!: An Experimental Study of Group Maintenance in Collaborative Writing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. ACM, New York, NY, USA, 961–970. https://doi.org/10.1145/2470654.2466123
[4] Tom Boellstorff, Bonnie Nardi, Celia Pearce, and T. L. Taylor. 2013. Words with Friends: Writing Collaboratively Online. *Interactions* 20, 5 (Sept. 2013), 58–61. https://doi.org/10.1145/2501987
[5] Christopher Bogart, Christian Kästner, James Herbsleb, and Ferdian Thung. 2016. How to break an API: cost negotiation and community values in three software ecosystems. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 109–120.
[6] Glenn A. Bowen. 2006. Grounded Theory and Sensitizing Concepts. *International Journal of Qualitative Methods* 5, 3 (2006), 12–23. https://doi.org/10.1177/160940690600500304
[7] Samridhi Choudhary, Christopher Bogart, Carolyn Rose, and Jim Herbsleb. 2020. Using Productive Collaboration Bursts to Analyze Open Source Collaboration Effectiveness. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 400–410.
[8] Juliet Corbin and Anselm Straus. 2014. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Thousand Oaks, CA, Sage.
[9] John W. Creswell and Dana L. Miller. 2000. Determining Validity in Qualitative Inquiry. *Theory Into Practice* 29, 3 (2000), 124–130.
[10] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social Coding in GitHub: Transparency and Collafboration in an Open Software Repository. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work* (Seattle, Washington, USA) *(CSCW '12)*. ACM, New York, NY, USA, 1277–1286. https://doi.org/10.1145/2145204.2145396
[11] Paul Dourish and Victoria Bellotti. 1992. Awareness and Coordination in Shared Workspaces. In *Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work* (Toronto, Ontario, Canada) *(CSCW '92)*. Association for Computing Machinery, New York, NY, USA, 107–114. https://doi.org/10.1145/143457.143468
[12] Nadia Eghbal. 2016. Roads and Bridges. *The Unseen labor behind our digital infrastructure* (2016). http://brochures.sisalp.fr/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure.pdf
[13] C. A. Ellis and S. J. Gibbs. 1989. Concurrency Control in Groupware Systems. *SIGMOD Rec.* 18, 2 (June 1989), 399–407. https://doi.org/10.1145/66926.66963
[14] Robert S. Fish, Robert E. Kraut, and Mary D. P. Leland. 1988. Quilt: A Collaborative Tool for Cooperative Writing. In *Proceedings of the ACM SIGOIS and IEEECS TC-OA 1988 Conference on Office Information Systems* (Palo Alto, California, USA) *(COCS '88)*. ACM, New York, NY, USA, 30–37. https://doi.org/10.1145/45410.45414
[15] Andrea Forte, Niki Kittur, Vanessa Larco, Haiyi Zhu, Amy Bruckman, and Robert E. Kraut. 2012. Coordination and Beyond: Social Functions of Groups in Open Content Production. In *Proceedings of the ACM 2012 Conference on*

*Computer Supported Cooperative Work* (Seattle, Washington, USA) *(CSCW '12)*. ACM, New York, NY, USA, 417–426. https://doi.org/10.1145/2145204.2145270

[16] Jolene Galegher and Robert E. Kraut. 1994. Computer-mediated communication for intellectual teamwork: An experiment in group writing. *Information systems research* 5, 2 (1994), 110–138.

[17] R Stuart Geiger, Nelle Varoquaux, Charlotte Mazel-Cabasse, and Chris Holdgraf. 2018. The types, roles, and practices of documentation in data analytics open source software libraries. *Computer Supported Cooperative Work (CSCW)* 27, 3-6 (2018), 767–802.

[18] Michael Gilbert, Jonathan T. Morgan, David W. McDonald, and Mark Zachry. 2013. Managing Complexity: Strategies for Group Awareness and Coordinated Action in Wikipedia. In *Proceedings of the 9th International Symposium on Open Collaboration* (Hong Kong, China) *(WikiSym '13)*. ACM, New York, NY, USA, 5:1–5:10. https://doi.org/10.1145/2491055.2491060

[19] Georgios Gousios, Margaret-Anne Storey, and Alberto Bacchelli. 2016. Work Practices and Challenges in Pull-Based Development: The Contributor's Perspective. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)* (Austin, TX, USA). 285–296. https://doi.org/10.1145/2884781.2884826

[20] Georgios Gousios, Andy Zaidman, Margaret-Anne Storey, and Arie van Deursen. 2015. Work Practices and Challenges in Pull-based Development: The Integrator's Perspective. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1* (Florence, Italy) *(ICSE '15)*. IEEE Press, Piscataway, NJ, USA, 358–368. http://dl.acm.org/citation.cfm?id=2818754.2818800

[21] James Herbsleb. 2016. Building a socio-technical theory of coordination: why and how (outstanding research award). In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 2–10.

[22] James D. Herbsleb and Audris Mockus. 2003. An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on software engineering* 29, 6 (2003), 481–494.

[23] Rubinetti Vincent Slochower David R. Hu Dongbo Malladi Venkat S. Greene Casey S. Himmelstein, Daniel S. and Anthony Gitter. 2019. Open collaborative writing with Manubot. *PLOS Computational Biology* 15, 6 (2019), e1007128. https://doi.org/10.1371/journal.pcbi.1007128

[24] Jing Jiang, David Lo, Jiahuan He, Xin Xia, Pavneet Singh Kochhar, and Li Zhang. 2017. Why and How Developers Fork What from Whom in GitHub. *Empirical Softw. Engg.* 22, 1 (Feb. 2017), 547–578. https://doi.org/10.1007/s10664-016-9436-6

[25] Hee-Cheol Ezra Kim and Kerstin Severinson Eklundh. 2001. Reviewing practices in collaborative writing. *Computer Supported Cooperative Work (CSCW)* 10, 2 (2001), 247–259. https://doi.org/10.1023/A:1011229212323

[26] Aniket Kittur and Robert E. Kraut. 2010. Beyond Wikipedia: Coordination and Conflict in Online Production Groups. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work* (Savannah, Georgia, USA) *(CSCW '10)*. ACM, New York, NY, USA, 215–224. https://doi.org/10.1145/1718918.1718959

[27] Meng-Fen Grace Lin, Suthiporn Sajjapanroj, and J. Curtis Bonk. 2011. Wikibooks and Wikibookians: Loosely Coupled Community or a Choice for Future Textbooks? *IEEE Transactions on Learning Technologies* 4, 4 (Oct. 2011), 327–339. https://doi.org/10.1109/TLT.2011.12

[28] Justin Longo and Tanya M. Kelley. 2015. Use of GitHub As a Platform for Open Collaboration on Text Documents. In *Proceedings of the 11th International Symposium on Open Collaboration* (San Francisco, California) *(OpenSym '15)*. ACM, New York, NY, USA, Article 22, 2 pages. https://doi.org/10.1145/2788993.2789838

[29] Justin Longo and Tanya M Kelley. 2016. GitHub use in public administration in Canada: Early experience with a new collaboration tool. *Canadian Public Administration* 59, 4 (2016), 598–623. https://doi.org/10.1111/capa.12192

[30] Thomas Malone and Kevin Crowston. 1994. The interdisciplinary study of coordination. *ACM Computing Surveys (CSUR)* 26, 1 (1994), 87–119.

[31] Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. 2013. Impression Formation in Online Peer Production: Activity Traces and Personal Profiles in Github. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work* (San Antonio, Texas, USA) *(CSCW '13)*. Association for Computing Machinery, New York, NY, USA, 117–128. https://doi.org/10.1145/2441776.2441792

[32] Ines Mergel. 2015. Open collaboration in the public sector: The case of social coding on GitHub. *Government Information Quarterly* 32, 4 (2015), 464–472. https://doi.org/10.1016/j.giq.2015.09.004

[33] Audris Mockus, Roy T. Fielding, and James Herbsleb. 2000. A Case Study of Open Source Software Development: The Apache Server. In *Proceedings of the 22Nd International Conference on Software Engineering* (Limerick, Ireland) *(ICSE '00)*. ACM, New York, NY, USA, 263–272. https://doi.org/10.1145/337180.337209

[34] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. 2002. Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Trans. Softw. Eng. Methodol.* 11, 3 (July 2002), 309–346. https://doi.org/10.1145/567793.567795

[35] Jonathan T. Morgan, Michael Gilbert, David W. McDonald, and Mark Zachry. 2014. Editing Beyond Articles: Diversity & Dynamics of Teamwork in Open Collaborations. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Baltimore, Maryland, USA) *(CSCW '14)*. ACM, New York, NY, USA, 550–563.

https://doi.org/10.1145/2531602.2531654

[36] Christine M. Neuwirth, David S. Kaufer, Ravinder Chandhok, and James H. Morris. 1990. Issues in the Design of Computer Support for Co-authoring and Commenting. In *Proceedings of the 1990 ACM Conference on Computer-supported Cooperative Work* (Los Angeles, California, USA) *(CSCW '90)*. ACM, New York, NY, USA, 183–195. https://doi.org/10.1145/99332.99354

[37] Kaufer David Chandhok Ravinder Neuwirth, Chris and Jim Morris. 2001. Computer support for distributed collaborative writing: A coordination science perspective. In *Coordination Theory and Collaboration Technology*. Lawrence Erlbaum Associates, New Jersey, 535–558.

[38] Judith S. Olson, Gary M. Olson, Marianne Storrøsten, and Mark Carter. 1993. Groupwork Close Up: A Comparison of the Group Design Process with and Without a Simple Group Editor. *ACM Trans. Inf. Syst.* 11, 4 (Oct. 1993), 321–348. https://doi.org/10.1145/159764.159763

[39] Judith S. Olson, Dakuo Wang, Gary M. Olson, and Jingwen Zhang. 2017. How People Write Together Now: Beginning the Investigation with Advanced Undergraduates in a Project Course. *ACM Trans. Comput.-Hum. Interact.* 24, 1, Article 4 (March 2017), 40 pages. https://doi.org/10.1145/3038919

[40] Kendall Powell. 2016. Does it take too long to publish research? *Nature* 530, 7589 (2016), 148–151. https://doi.org/10.1038/530148a

[41] Carola Strobl. 2014. Affordances of Web 2.0 technologies for collaborative advanced writing in a foreign language. *Calico Journal* 31, 1 (2014), 1–19.

[42] Yunting Sun, Diane Lambert, Makoto Uchida, and Nicolas Remy. 2014. Collaboration in the Cloud at Google. In *Proceedings of the 2014 ACM Conference on Web Science* (Bloomington, Indiana, USA) *(WebSci '14)*. Association for Computing Machinery, New York, NY, USA, 239–240. https://doi.org/10.1145/2615569.2615637

[43] Bill Tomlinson, Joel Ross, Paul Andre, Eric Baumer, Donald Patterson, Joseph Corneli, Martin Mahaux, Syavash Nobarany, Marco Lazzari, Birgit Penzenstadler, Andrew Torrance, David Callele, Gary Olson, Six Silberman, Marcus Stünder, Fabio Romancini Palamedi, Albert Ali Salah, Eric Morrill, Xavier Franch, Florian Floyd Mueller, Joseph 'Jofish' Kaye, Rebecca W. Black, Marisa L. Cohn, Patrick C. Shih, Johanna Brewer, Nitesh Goyal, Pirjo Näkki, Jeff Huang, Nilufar Baghaei, and Craig Saper. 2012. Massively Distributed Authorship of Academic Papers. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI EA '12)*. ACM, New York, NY, USA, 11–20. https://doi.org/10.1145/2212776.2212779

[44] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Influence of Social and Technical Factors for Evaluating Contribution in GitHub. In *Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE 2014)*. Association for Computing Machinery, New York, NY, USA, 356–366. https://doi.org/10.1145/2568225.2568315

[45] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Let's Talk about It: Evaluating Contributions through Discussion in GitHub. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Hong Kong, China) *(FSE 2014)*. Association for Computing Machinery, New York, NY, USA, 144–154. https://doi.org/10.1145/2635868.2635882

[46] Andrew H Van de Ven, Andre L Delbecq, and Richard Koenig Jr. 1976. Determinants of coordination modes within organizations. *American sociological review* (1976), 322–338. https://doi.org/10.2307/2094477

[47] Andrew H Van de Ven, Andre L Delbecq, and Richard Koenig Jr. 1976. Determinants of coordination modes within organizations. *American sociological review* (1976), 322–338. https://doi.org/10.2307/2094477

[48] Dakuo Wang, Haodan Tan, and Tun Lu. 2017. Why Users Do Not Want to Write Together When They Are Writing Together: Users' Rationales for Today's Collaborative Writing Practices. *Proc. ACM Hum.-Comput. Interact.* 1, CSCW, Article 107 (Dec. 2017), 18 pages. https://doi.org/10.1145/3134742

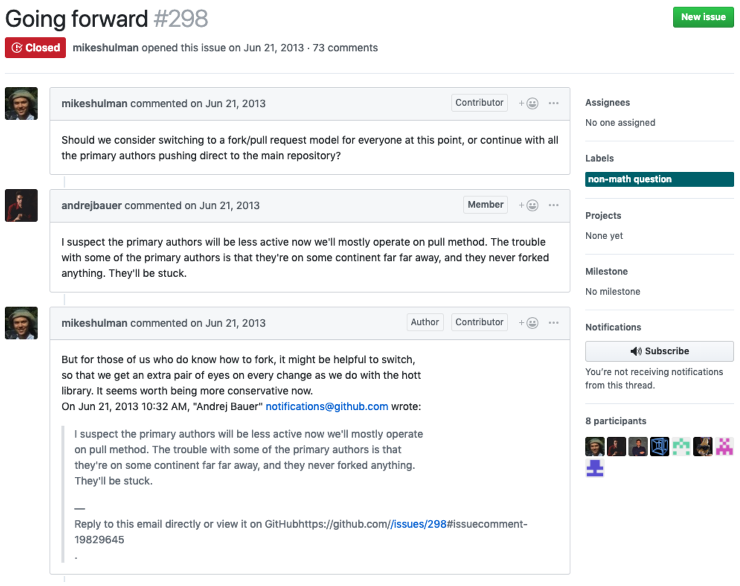
[49] Robert K Yin. 2014. *Case study research and applications: Design and methods.* Sage publications.

[50] Alexey Zagalsky, Joseph Feliciano, Margaret-Anne Storey, Yiyun Zhao, and Weiliang Wang. 2015. The Emergence of GitHub As a Collaborative Platform for Education. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Vancouver, BC, Canada) *(CSCW '15)*. ACM, New York, NY, USA, 1906–1917. https://doi.org/10.1145/2675133.2675284

[51] Vasilescu Bogdan Zhou, Shurui and Christian Kästner. 2020. How has forking changed in the last 20 years? a study of hard forks on GitHub. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE 2020)*. 445–456.

## A APPENDIX: REFERENCE CASE EVENTS

This appendix contains images associated with key case events described in our analysis and reporting. These events were publicly visible on GitHub itself and representative of the types of data included in our analysis of archival data for each case.



mikeshulman opened this issue on Jun 21, 2013 · 73 comments

mikeshulman commented on Jun 21, 2013

Should we consider switching to a fork/pull request model for everyone at this point, or continue with all the primary authors pushing direct to the main repository?

andrejbauer commented on Jun 21, 2013

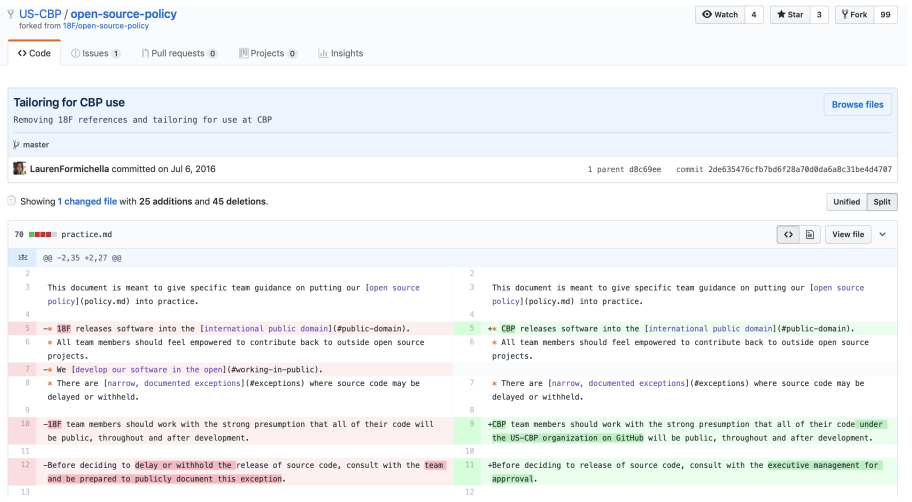
I suspect the primary authors will be less active now we'll mostly operate on pull method. The trouble with some of the primary authors is that they're on some continent far far away, and they never forked anything. They'll be stuck.

mikeshulman commented on Jun 21, 2013

But for those of us who do know how to fork, it might be helpful to switch, so that we get an extra pair of eyes on every change as we do with the hott library. It seems worth being more conservative now.
On Jun 21, 2013 10:32 AM, "Andrej Bauer" notifications@github.com wrote:

> I suspect the primary authors will be less active now we'll mostly operate on pull method. The trouble with some of the primary authors is that they're on some continent far far away, and they never forked anything. They'll be stuck.
>
> —
>
> Reply to this email directly or view it on GitHubhttps://github.com//issues/298#issuecomment-19829645
> .

Fig. 4. A screenshot of the decision to move to a "fork and pull-request" model on the HoTT project, meaning that all changes would be made in external versions of the repository and submitted via pull-request rather than being modified directly.



Tailoring for CBP use

Removing 18F references and tailoring for use at CBP

LaurenFormichella committed on Jul 6, 2016 1 parent d8c69ee commit 2de635476cfb7bd6f28a70d0da6a8c31be4d4707

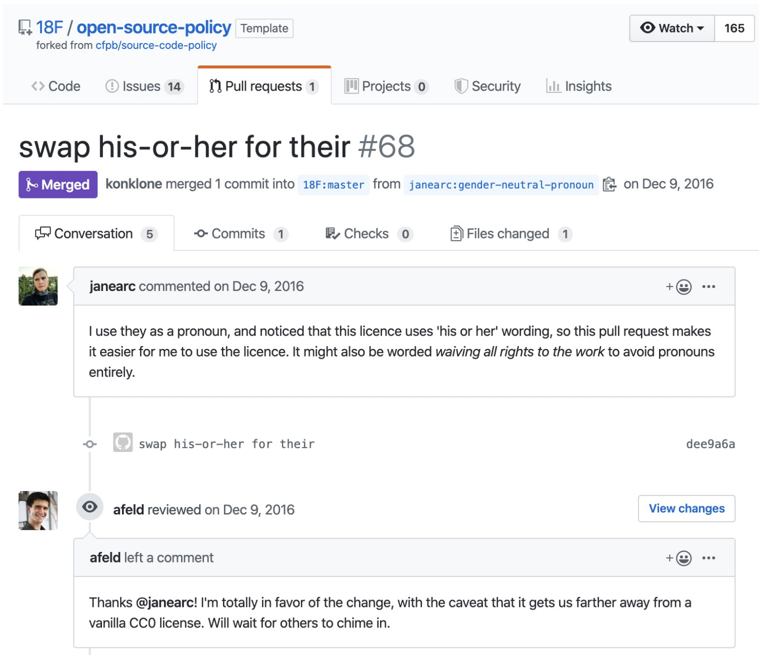
Showing 1 changed file with 25 additions and 45 deletions.

70 practice.md

```
@@ -2,35 +2,27 @@

 This document is meant to give specific team guidance on putting our [open source
 policy](policy.md) into practice.

-* 18F releases software into the [international public domain](#public-domain).
 * All team members should feel empowered to contribute back to outside open source
 projects.
-* We [develop our software in the open](#working-in-public).
 * There are [narrow, documented exceptions](#exceptions) where source code may be
 delayed or withheld.

-18F team members should work with the strong presumption that all of their code will
 be public, throughout and after development.

-Before deciding to delay or withhold the release of source code, consult with the team
 and be prepared to publicly document this exception.

```

```

 This document is meant to give specific team guidance on putting our [open source
 policy](policy.md) into practice.

+* CBP releases software into the [international public domain](#public-domain).
 * All team members should feel empowered to contribute back to outside open source
 projects.

 * There are [narrow, documented exceptions](#exceptions) where source code may be
 delayed or withheld.

+CBP team members should work with the strong presumption that all of their code under
 the US-CBP organization on GitHub will be public, throughout and after development.

+Before deciding to release of source code, consult with the executive management for
 apprroval.

```

Fig. 5. U.S. Customs and Border Protection tailoring the 18F open source policy for the agency needs


18F / open-source-policy Template
forked from cfpb/source-code-policy
Watch 165
Code
Issues 14
Pull requests 1
Projects 0
Security
Insights

swap his-or-her for their #68

Merged konklone merged 1 commit into 18F:master from janearc:gender-neutral-pronoun on Dec 9, 2016

janearc commented on Dec 9, 2016

I use they as a pronoun, and noticed that this licence uses 'his or her' wording, so this pull request makes it easier for me to use the licence. It might also be worded *waiving all rights to the work* to avoid pronouns entirely.

swap his-or-her for their dee9a6a

afeld reviewed on Dec 9, 2016

afeld left a comment

Thanks @janearc! I'm totally in favor of the change, with the caveat that it gets us farther away from a vanilla CC0 license. Will wait for others to chime in.

Fig. 6. A pull request updating the language of policy toward more gender neutrality

## B APPENDIX: CASE ANALYSIS RESULTS

This appendix contains tables representing the results of our within and across case analysis. Table 6 presents the types of contributions the HoTT book project repository received on GitHub. Table 7 presents the types of contributions the 18F open source policy received on GitHub. Finally, Table 8 contrasts our two cases in terms of collaboration mode and related case features.

Received June 2020; revised October 2020; accepted December 2020

Table 6. A Summary of HoTT book contributions on GitHub

| Contribution types | | Description | Examples |
|---|---|---|---|
| Content | Minor fixes | Minor math errors; math and text typos; clarification; global auto corrects | “Removed an unnecessary restriction in exercise 7.5” (PR #518)<br>“Clarify subset complement comment” (PR #511)<br>“Improve wording of ‘that that‘ fix” (PR #475)<br>“Change the pictures to palette, run some through optipng and/or pngout.” (PR #251) |
| | Substantive fixes | Fixes to theorem; new math content | "Thm2.11.4" (PR #502)<br>"An exercise relating embeddings to left-cancellable maps" (PR #686)<br>"Solutions to Exercises 1.1 to 1.3" (PR #436) |
| | Presentation fixes | Formatting; LaTex fixes | "Consistent ordering: 0_2 then 1_2" (PR #375)<br>"Correct a whitespace error" (PR #354)<br>"Some inappropriate uses of \text to \math" (PR #422)<br>"Fix an overfull hbox in the proof of Theorem 7.2.2" (PR #421) |
| Process | Process change | Switching to new production model (e.g., fork/pull request model); continuous integration; automated errata; auto-create up-to-the-minute version of the book | "Going forward - Should we consider switching to a fork/pull request model for everyone at this point, or continue with all the primary authors pushing direct to the main repository?" (Issue #298)<br>"Do you want a Travis-CI instance?" (PR #540)<br>"Make a list of errata" (PR #328) |
| | Infrastructure maintenance | Fixing tools (e.g., replacing library/package) | "Fix the nightly build script" (PR #825)<br>"A script to filter the errata" (PR #426)<br>"A script to automatically check whether label numbers have changed" (PR #371) |

Table 7. A summary of 18F contributions on GitHub

| Contribution types | Description | Examples |
|---|---|---|
| Improving accessibility | Make the project and associated communication documents more accessible | “Adding some friendly introduction text” (PR #19)<br><br>“I updated ‘CONTRIBUTING.md’ to link to our Code of Conduct. I also rearranged and rewrote some of the text, so that the welcome section was smaller and potentially less intimidating (especially since we’re mentioning a set of rules now)” (PR #39) |
| Extending policy | Extending the policy itself through more general and concise wording | “I use ‘they’ as a pronoun, and noticed that this licence uses ’his or her’ wording, so this pull request makes it easier for me to use the licence. It might also be worded *waiving all rights to the work* to avoid pronouns entirely” (PR #68)<br><br>Suggested to use “‘terms permitted by’ rather than ‘in terms of’, to state more precisely the licensing implications for joint works, and rephrase to affirm that all open source licenses permit internal use without redistribution, rather than implying that only some OSS licenses do.” (PR #15) |
| Clarifying project process | Changes and requests for change were associated with explaining or clarifying process norms | “On practices page, updating #compliance to #infrastructure” (PR #62)<br><br>“Add information to make clear that our repos are iterative and what people see may not be a final version” (PR #52) |
| Clarifying OSS process | Changes and requests for change were related to how work is done in open source software more broadly | “Add paragraph to team practices doc encouraging upstream contributions” (PR #22)<br><br>“Add section to practices doc about accepting contributions” (PR #29)<br><br>“FOSS vs OSS” (PR #43) |

Table 8. Case comparison: The HoTT book and 18F open source policy document - perfection vs. evolution styles of open pull-based writing collaboration

| | | HoTT Textbook | 18F Policy Document |
|---|---|---|---|
| Collaboration | | **Perfecting** | **Evolution** |
| | | Fork-and-pull model used to make modifications that correct errors, extend knowledge, provide additional examples for application | Fork-and-pull model used to adopt document and customize for local scope |
| Artifact | | **Ground truth** | **Living document** |
| | Type | Knowledge object | Policy |
| | Lifecycle | Single book, future editions | Successive forking and adaptation |
| | Synchronization | Manage print and digital versions | Monitor upstream and downstream projects for useful changes |
| | Content deliberation | Social epistemology | Policy negotiation, clarification |
| Openness | | **Optimizing** | **Customizing** |
| | Intention | Soliciting contributions to perfect artifact | Signaling openness in software collaboration |
| | Audience | Software savvy mathematicians | Government agencies (18F, clients and collaborating organizations) |
| | Scope | Community of practice (Mathematicians studying homotopy theory) | Organization adopting the policy |