GitHub Repository Complexity Leads to Diminished Web Archive Availability

David Calano
Old Dominion University
Department of Computer Science
Norfolk, Virginia 23529, USA

Michele C. Weigle
Old Dominion University
Department of Computer Science
Norfolk, Virginia 23529, USA

Michael L. Nelson
Old Dominion University
Department of Computer Science
Norfolk, Virginia 23529, USA

ABSTRACT

Software is often developed using versioned controlled software, such as Git, and hosted on centralized Web hosts, such as GitHub and GitLab. These Web hosted software repositories are made available to users in the form of traditional HTML Web pages for each source file and directory, as well as a presentational home page and various descriptive pages. We examined more than 12,000 Web hosted Git repository project home pages, primarily from GitHub, to measure how well their presentational components are preserved in the Internet Archive, as well as the source trees of the collected GitHub repositories to assess the extent to which their source code has been preserved. We found that more than 31% of the archived repository home pages examined exhibited some form of minor page damage and 1.6% exhibited major page damage. We also found that of the source trees analyzed, less than 5% of their source files were archived, on average, with the majority of repositories not having source files saved in the Internet Archive at all. The highest concentration of archived source files available were those linked directly from repositories' home pages at a rate of 14.89% across all available repositories and sharply dropping off at deeper levels of a repository's directory tree.

CCS CONCEPTS

• **Information systems** → **World Wide Web**; • **Computer systems organization** → *Reliability*.

KEYWORDS

World Wide Web, Web Archives, Git, GitHub, Software, Software Repositories, Digital Preservation, Memento, Memento Damage

ACM Reference Format:
David Calano, Michele C. Weigle, and Michael L. Nelson. 2025. GitHub Repository Complexity Leads to Diminished Web Archive Availability. In *17th ACM Web Science Conference (Websci '25), May 20–24, 2025, New Brunswick, NJ, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3717867.3717920

1 INTRODUCTION

Modern software projects are powered by version control software, such as Git,[1] that enable an inherent level of preservation due to their distributed nature. The Git hosting platform GitHub quickly rose to prominence in 2008 as one of the first centralized Git Web hosts, providing a Web interface to what is otherwise just a collection of files in a directory on a user's local machine. Web resources in GitHub are a superset of the files in Git, with GitHub providing "Issues", "Discussions", "Actions", and other pages that have no analog in Git itself.

An issue with such centralization is that should something happen to an individual Web hosted repository or the entire Web host itself, such as corruption, deletion, or a migration from open to closed source, the hosted project's availability would end for any programmer, academic, or researcher who might need access to it. The only remaining versions available in such cases would be to access preserved versions from various Web and software archiving services. Although dedicated archives, such as Software Heritage,[2] exist that mirror and directly preserve publicly available source code, they should not be exclusively relied upon. The dynamic structure of utilizing numerous individual Web pages to represent a repository's source content can often cause problems for traditional Web archiving systems. This characteristic representation often leads to a lack of availability with respect to an archived repository's content and can result in the inability to compile or recreate and utilize software artifacts derived from archived source code. Given the importance of software repositories across the digital ecosystem, we sought to determine how well repositories are being preserved should they no longer be available on the live Web.

To assess the fidelity of archived software repositories, we explore the extent to which hosted software repositories from GitHub,[3] and additionally from various centralized software hosting platforms, such as GitLab,[4] Sourceforge,[5] and BitBucket,[6] were archived within the Internet Archive's Wayback Machine. Motivated by our past work [18] that measured citations to Git archives in scholarly papers, we now measure how well archived is the archived source tree content. Our goal is to ascertain the accuracy to which software repositories are being archived to the Wayback Machine, as measured by their amount of page damage, as well as to assess the extent to which the source code of Web hosted software repositories are being archived by Web archives. The ultimate aim is to determine if a Web-hosted Git project that has been archived to the Internet Archive's Wayback Machine could successfully be rebuilt using only the archived content. Such a method would be one of the only viable options in the scenario where a project ceases to exist on the live Web and has failed to be preserved in other archives. To

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

ACM ISBN 979-8-4007-1483-2/2025/05.
https://doi.org/10.1145/3717867.3717920

[2]https://www.softwareheritage.org/
[3]https://github.com
[4]https://gitlab.com
[5]https://sourceforge.net
[6]https://bitbucket.com

achieve these goals, we examined both the presentational pages for archived repositories as well as their archived source trees. Through our examination of archived software repositories, we find that virtually all analyzed source trees archived in the Wayback Machine contain a degree of missing content, leaving them unable to be reconstructed using their archived Web page representations of the source code.

2 BACKGROUND

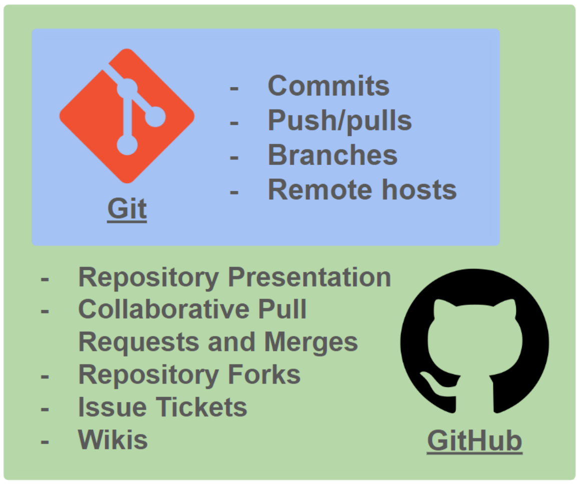


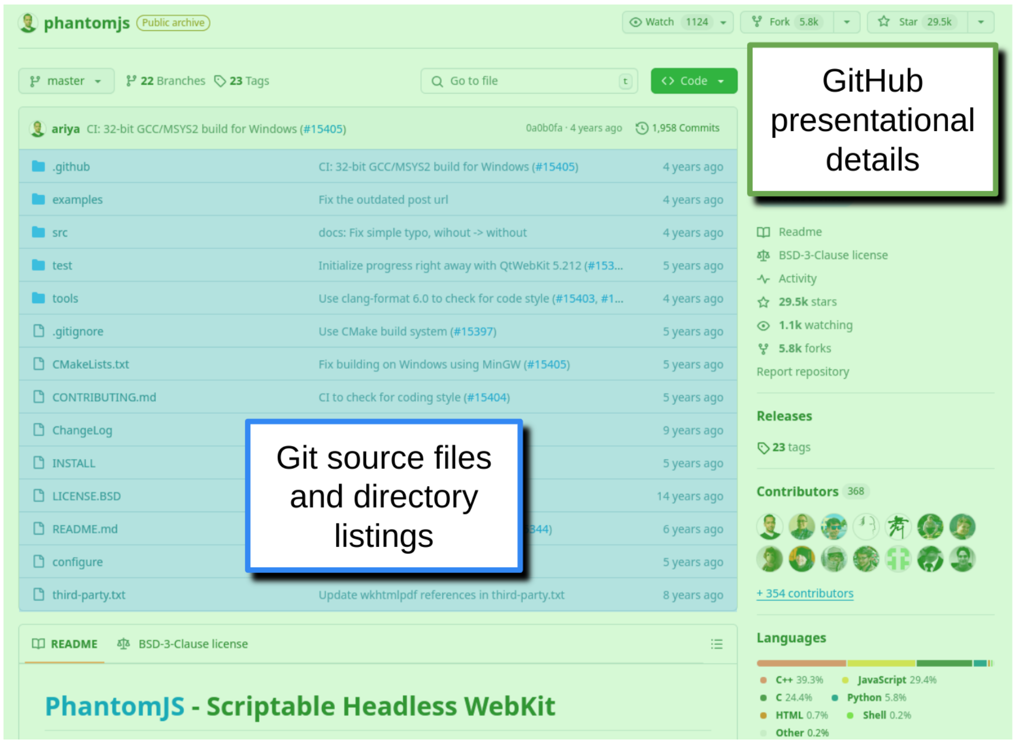
Figure 1: Breakdown of the differences between the version control software Git and the Git hosting platform GitHub.

Version control software, such as Git, acts as a distributed system for developing software stored as a repository of source files and directories. These distributed Git repositories can be linked to one or more Web servers that act as centralized views of these inherently distributed version control systems that can enhance the set of features provided by Git itself, described in Figure 1. The websites where Git repositories are hosted typically also provide a collection of HTML pages that act as a public portal by which Web users can view and interact with the repository. Numerous websites provide online hosting services for these Git repositories, such as Sourceforge, which was previously popular among software developers as a hub to post source code and packaged binary releases. Shortly after its launch in 2008 GitHub began ramping up in popularity as the dominant platform with its focus on community and social development [20]. Other alternatives exist as well, such as BitBucket, provided by thMaintaining the Integrity of the Specificationse company Atlassian and integrating with their business-focused development ecosystem, and GitLab, which has also become a community favorite thanks to a rich feature set, open source focus, and the ability for developers to self-host their own versions of the platform.

The software projects hosted on these Web platforms are typically bifurcated into two components, the source files and directories themselves and the related presentational content, such as the rendered README file, issue discussion boards, repository metadata and contributors, and hosted binary release page. This is depicted in Figure 2, with the page area highlighted in blue showing the table listing and linking to the repository's top-level source directories and files and the surrounding green area describing the external, associated content generated by GitHub to describe the repository. These presentational pages include the project's landing page rendering a descriptive README and overview of the project, project-specific wiki pages, an index of project release versions, pages detailing historical code tests, and Web pages for issues and pull requests that typically take on the form of discussion boards. Depending upon the Web site where the project is hosted, there may be additional pages detailing the project's code composition, popularity metrics, indexes of contributors and remixed project versions, further non-issue discussions, and other information related to project activity. A repository's README is a special case here, as it is an optional (though generally present) text or Markdown-formatted file that is included within the repository alongside other source files but is almost always rendered into the project's presentational landing page and presented to the user as a traditional HTML Web page. These presentational and descriptive Web pages are not themselves part of the Git repository but are derived from it and then stored separately in another location, such as an external database managed by the Web host. Each source file for a Web hosted repository has two versions, an HTML formatted page that has the source content embedded within it and a raw text version. The HTML page embedded with the source code (example in Figure 3a) is the default page that a user sees when browsing a repository from the Web hosting platform. The raw version is accessed through a button on the source's HTML page and is usually hosted under a different domain or subdomain. For GitHub, the default HTML pages for source files are presented through the normal github.comdomain, while raw versions are served through the raw.githubusercontent.com domain and exist under a slightly different URI path.

With software having such a driving importance in our digital lives and increasingly in academic research [24], the preservation of software is a contributor to the issue of academic reproducibility, commonly referenced as a "replication crisis" [16, 23, 39]. Open source software and data are important components in cultivating a more open methodology of science to alleviate this crisis [8, 9, 22, 38], though software is useless if it cannot be found or accessed [28]. There have been multiple attempts at creating alternative access systems and dedicated archives for Git-based content that are now defunct, such as GHTorrent,[7] or no longer updated and maintained, such as the Public Git Archive project [30] and GHArchive.[8] In the case of GHTorrent, their provided Web site has now become the home for junkware. Smaller projects and hosts are not the only ones subject to shuttering either, as noted by GitHub's rise as the predominant repository host overtaking the popularity of previous platforms, such as Sourceforge. Another example of this is the closing of the Google Code platform in 2016 [15] that left its over 1.5 million hosted projects no option but to close or migrate to other platforms. While development on this platform is no longer possible, Google has left the code and issues themselves accessible through a static, archived portal. The Google Code platform was

[7]https://github.com/ghtorrent/
[8]https://www.gharchive.org/



ariya CI: 32-bit GCC/MSYS2 build for Windows (#15405) 0a0b0fa · 4 years ago 1,958 Commits

.github	CI: 32-bit GCC/MSYS2 build for Windows (#15405)	4 years ago
examples	Fix the outdated post url	4 years ago
src	docs: Fix simple typo, wihout -> without	4 years ago
test	Initialize progress right away with QtWebKit 5.212 (#153...	5 years ago
tools	Use clang-format 6.0 to check for code style (#15403, #1...	4 years ago
.gitignore	Use CMake build system (#15397)	5 years ago
CMakeLists.txt	Fix building on Windows using MinGW (#15405)	5 years ago
CONTRIBUTING.md	CI to check for coding style (#15404)	5 years ago
ChangeLog		9 years ago
INSTALL		5 years ago
LICENSE.BSD		14 years ago
README.md	344)	6 years ago
configure		5 years ago
third-party.txt	Update wkhtmlpdf references in third-party.txt	8 years ago

PhantomJS - Scriptable Headless WebKit

Figure 2: Screenshot of the GitHub home page for the PhantomJS repository showing the embedded representation of the project's top-level source code in blue and the presentational elements of the project generated by GitHub in green.

once one of the most popular platforms for Web hosted software repositories. The shuttering of such a widely used platform serves to further highlight the fragile ecosystem of software archiving and the importance of supporting and maintaining those still operational, such as GitHub's Archive Program, Software Heritage, and archiving institutions such as the Internet Archive. It is important to have a variety of archives available as the centralization and concentration of archived content within a one institution can cause it to become a single point of failure. Should any centralized institution become the exclusive source of truth for archived content then that content runs the risk of becoming unavailable should that source itself become unavailable.

Software Heritage is a UNESCO-supported, non-profit initiative aiming to preserve all publicly available source code. Software Heritage focuses specifically on archiving source code and currently holds the largest existing public collection of source code. To date, the archive has preserved the source code for over 300 million software projects collected from 33 major forges, a common term for Git hosting platforms. While smaller, publicly available repository Web hosts might still be overlooked and missed by Software Heritage, they do allow for user requests to archive arbitrary Web repository hosts so they may also be preserved. Despite Software Heritage extensively collecting publicly available source code [36], it does not archive related pages found on software Web hosts, such as the framed HTML pages in which source files are embedded or any project page directly, such as its landing page or issues page, nor does it seek to directly archive binary releases. It is not a direct copy or mirror of Web based Git platforms but collects source code from them as unique digital artifacts for long-term preservation. In contrast, the Internet Archive is more general, primarily preserving Web pages through its Wayback Machine archive, but also containing archives for books, film, audio, and software. The Internet Archive's software archive[9] is not equivalent to that of Software Heritage in that while Software Heritage focuses on public source code, the Internet Archive's software archive is oriented towards more packaged releases of digital software code and extracted optical disk images.

3 RELATED WORK

The interplay between software and scholars has been extensively researched. One of the leading groups on this subject is the Investigating & Archiving the Scholarly Git Experience project.[10] Their research has looked at the importance of preserving software [43], the behavior studies of scholars writing code [34], the usage patterns of scholars working with Git platforms [32], and the challenges faced by scholars and their use of Git [33]. Their work helps to denote how software is utilized and the importance of its preservation outside of software development in academia and library sciences [42, 44]. This importance goes both ways as well [46], with the development of software often relying on academic research for foundational knowledge, algorithms, and data. With such a strong relationship between software and academic

[9]https://archive.org/details/software
[10]https://investigating-archiving-git.gitlab.io/

research and our limited ability to trust the content linked from academic publications upon future review, as noted by Jones et al. [25], strong Web archiving systems must be developed, similar to LOCKSS [29] and CLOCKSS [37]. With the lack of a widespread, distributed network of repository mirrors, this burden heavily rests on efforts by groups such as Software Heritage and the content available in the Internet Archive's Wayback Machine.

Software Heritage is designed to be a universal archive for software source code [1, 10, 13, 40]. Software Heritage recognizes the aggregated source code from a software repository as important digital objects [12], assigning each collected repository with a unique identifier [11] to aid in their handling and use for scientific reproducibility. Their vast collection of archived software source code forms a rich data set that is utilized by researchers in multiple areas of study [35]. This dataset serves as the basis for numerous other research works related to open-source software and data mining [3] and as well as helping to further promote an atmosphere of open science [9, 14].

Despite Software Heritage's extensive catalog of preserved repositories, we have previously found that only 68.39% of analyzed repositories had been archived there [18]. We also explored the expanding prevalence of software repositories cited among scholarly publications from arXiv and PubMed Central [17] and described the tendencies of academic archives to focus on paper content but not associated source code [19]. Our findings showed a rise not only in general URIs referenced among academic papers but a steady increase in the number of references to Web hosted repositories from 2007 to 2021. Across a corpus of more than 11,000 PubMed Central papers, Sourceforge references were originally the most popular, tapering off and ultimately superseded by GitHub as the predominately cited Git hosting platform. For both the PubMed Central corpus and a corpus of over 125,000 arXiv papers, beginning around 2012 the citations to GitHub repositories grew continuously, with repositories hosted on GitHub being cited an average of around 20% of papers in the arXiv corpus by 2021. Concerning repositories archived in the Wayback Machine, we specifically looked at the presence of repository home pages. We found that despite a growing citation of repositories in scholarly work, around 7.44% of hosted repository URIs were no longer available on the live Web. Our research in this paper expands this by examining page content and the completeness of archived source code.

The Memento Damage project [41] aims to provide an estimate of damage present in an archived Web page. Damage in an archived Web page may be caused by content that was not captured when the original page was archived or issues arising from the archived page's replay that affect the final rendering of the page to the viewer. This includes missing page resources such as image or video elements, page stylesheets, and failed network requests made by JavaScript code responsible for the further loading of content within a page [6]. The original algorithm calculates a total page damage metric between 0% and 100% based on the number of missing page components that are weighted depending on their size, location, and determined importance within the page. This measurement is meant to be performed on archived Web pages, or mementos [45], but can be conducted for any Web page. This research was later expanded by looking at issues related to page sub-resources being loaded dynamically by JavaScript [5]. Other work peripheral to the Memento Damage project includes that of Kelly et al. [26] and Banos et al. [2], which seek to better evaluate the archivability of Web pages, as well as Kiesel et al. [27], working to evaluate the quality of archived Web pages through the comparison of archived and live Web pages. These works help to increase the reliability and accuracy of quality and damage measurements for archived Web pages when assessing the extent to which they were archived.

During our analysis of GitLab and BitBucket pages, we encountered issues related to the use of JavaScript for loading both partial and complete page content. The utilization of JavaScript for page loading can have potentially far-reaching consequences for Web archiving systems. This phenomenon has been found in other research as well, particularly in archived social media pages, as noted by Garg et al. [21] when researching archived Twitter pages and Bragg et al. [4] for archived Instagram pages. The impact of JavaScript and its relationship to Web archiving is also detailed by Brunelle et al. [7] and Weigle et al. [47].

4 METHODOLOGY

To analyze the **presentational components** of Web hosted repositories, we primarily selected over 10,000 archived public repository pages from GitHub and secondarily 1,000 from GitLab, 600 from BitBucket, and 500 from Sourceforge. The majority of GitHub repository URIs were pulled from data[11] that we previously collected [19]. Supplemental URIs were derived from manual searches within each repository host. We used https://github.com/topics/ for GitHub searches, https://sourceforge.net/directory/ for Sourceforge, https://gitlab.com/explore/projects/ for GitLab, and https://bitbucket.org/repo/all/ for BitBucket. To conduct these searches we looked for top and trending repositories within the various platform search pages using keywords such as "education", "data science", "web", and other similar keywords.[12] In order to assess the page damage of archived repository pages we utilized the Memento Damage tool to visit each URI, retrieve its HTML content, if available, and then assess the condition of the project's home page.

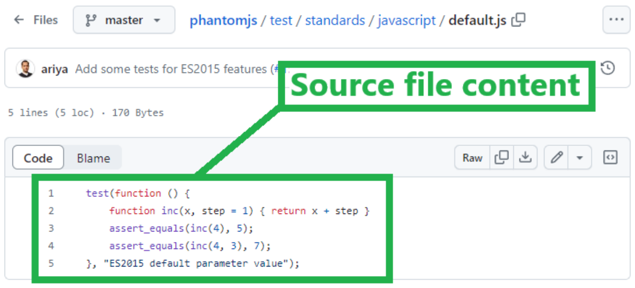
The assessment of an archived software project's **source tree** is a more extensive process, requiring the analysis of numerous pages, and was carried out separately from the presentational analysis. To perform this task we extended the capabilities of the Memento Damage tool with a new crawler component[13] capable of crawling and assessing the state of an archived software project from the Wayback Machine. In this paper, we also report on the archived source trees of hosted software repositories collected from GitHub that we found in the Wayback Machine. We chose to look at archived repositories from GitHub due to both the overall popularity of the platform and issues stemming from other Web hosts, described in the next section. To determine the extent of a project's archived source tree we first utilized GitHub's API and the Wayback Machine's CDX API[14] to get a record of the files in the repository's original source tree, their associated Web page representations and raw file URIs, and then their available mementos in the Wayback Machine. For our analysis we targeted the most recent memento

[11]https://github.com/oduwsdl/Extract-URLs
[12]https://gist.github.com/dcalano/15b61a7b0e2054fcf96a94ee6d317651
[13]https://github.com/oduwsdl/web-memento-damage/tree/code-crawler
[14]https://github.com/internetarchive/wayback/tree/master/wayback-cdx-server

with a 200 HTTP status code for each available source file, filtering out older versions for the same source file page, and then conducted page crawls using the Memento Damage tool to examine its content and ensure the page content was properly archived. For these archived source code Web pages, the damage metric is distinct from a traditional page analysis using the Memento Damage tool as any amount of damage to JavaScript, images, stylesheets, or other elements is acceptable, so long as the text for the source file is present on the page. The archived pages that were collected were then separately cross-referenced with the project's original pages from the live Web to determine the completeness of the archived source tree. While we recorded the complete archived source tree, for statistical measurements we chose to utilize the Web pages for source files, shown in Figure 3a, and not those of archived directories, shown in Figure 3b, within a repository. This is primarily because the content of source files serves as the key element of importance while the Web pages for directories merely serve as indexes linking to other directories and files. Since we can retrieve an overview of available pages from the Wayback Machine's CDX API, we do not need to crawl through directory index pages. In fact, because a project's directory structure can be derived using the source file paths present as part of their URIs, we deemed the directory pages themselves as irrelevant for analysis. While we are able to see the overview of available repository pages in the Wayback Machine, this does not mean we have a complete picture of the archived repository without external comparison to the original source tree. From a view of only the archived content, if a directory page was not archived, then the nested directory pages and source file pages linked within it would also likely be missing and unaccounted for within the Wayback Machine. The exception here would be nested files directly archived through manual submission or other means, circumventing an automated crawl. By analyzing the URI path of such an archived page, we can start to fill in the gaps and make note of a parent directory that is unavailable within the archived source tree, but we would not have visibility of any neighboring files from that missing directory.

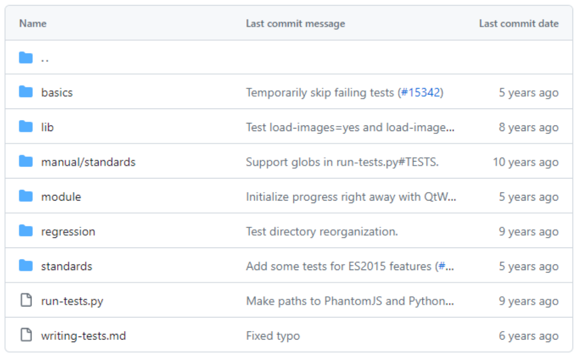
5 FINDINGS

Our findings reveal a general lack of completeness among archived repositories. These shortcomings are both due to the size and structuring of Web hosted software projects but also the architecture and traditional methodologies of Web archiving systems. An increased measure of captured page fidelity could be reached through general improvements in existing archival systems. For repository source trees though, the development of new crawlers or improved logic and behavioral patterns specifically designed to address source trees will be necessary to achieve an increase in the completeness of archived content. The practice of Web hosted Git platforms converting a repository source tree into many separate HTML pages is not a good match for Web crawlers or archive systems because, while many pages might be archived, it might only take a single missing source file page to invalidate the underlying source tree's integrity with respect to its ability to be reconstituted. The chance of such a compromise goes up as a function of the number of source files in a repository.


Files
master
phantomjs / test / standards / javascript / default.js
ariya Add some tests for ES2015 features (#
Source file content
5 lines (5 loc) · 170 Bytes
Code
Blame
Raw

```
test(function () {
    function inc(x, step = 1) { return x + step }
    assert_equals(inc(4), 5);
    assert_equals(inc(4, 3), 7);
}, "ES2015 default parameter value");
```

(a) Cropped screenshot of the Web page for a GitHub project source file https://github.com/ariya/phantomjs/blob/master/test/standards/javascript/default.js.



Name	Last commit message	Last commit date
..		
basics	Temporarily skip failing tests (#15342)	5 years ago
lib	Test load-images=yes and load-image...	8 years ago
manual/standards	Support globs in run-tests.py#TESTS.	10 years ago
module	Initialize progress right away with QtW...	5 years ago
regression	Test directory reorganization.	9 years ago
standards	Add some tests for ES2015 features (#...	5 years ago
run-tests.py	Make paths to PhantomJS and Python...	9 years ago
writing-tests.md	Fixed typo	6 years ago

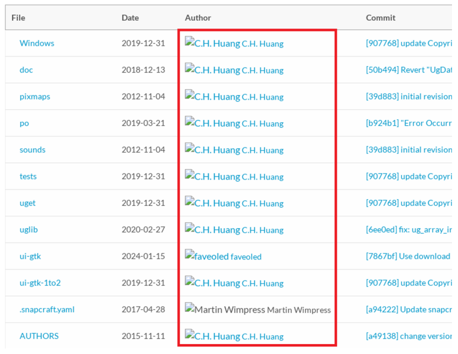
(b) Cropped screenshot of the Web page for a GitHub project source directory, https://github.com/ariya/phantomjs/tree/master/test, indexing further directories and files.

Figure 3: Cropped screenshots of directory and source file Web pages for GitHub repository.

5.1 Presentation Pages

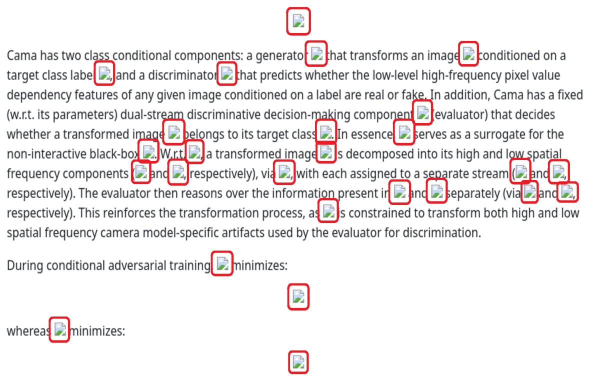
For the presentational component of our analysis, we found that 31.21% of GitHub repositories exhibit some form of damage, with around 1.43% having some form of significant damage to their archived home page when rendered upon playback. Collectively between GitHub and the secondary repository hosting platforms analyzed, around 10.33% of the sampled repository URIs did not have any mementos in the Wayback Machine. Of those remaining with successful captures, Table 1 shows that 58.22% exhibited no damage to their home page, 29.85% exhibited minor damage (defined here as page damage totaling less than 25%), and 1.6% exhibited major damage (25% and above), as measured by the Memento Damage tool. An example of minor damage might be a page[15] missing a few small or unimportant images, as depicted in Figure 4a, while those exhibiting significant damage[16] had more substantive problems such as large and/or numerous missing images or missing core

[15]http://web.archive.org/web/20240211225434/https://sourceforge.net/p/urlget/uget2/ci/master/tree/
[16]http://web.archive.org/web/20231114065050/https://github.com/jeroneandrews-sony/cama



File	Date	Author	Commit
Windows	2019-12-31	C.H. Huang C.H. Huang	[907768] update Copyri
doc	2018-12-13	C.H. Huang C.H. Huang	[50b494] Revert "UgDat
pixmaps	2012-11-04	C.H. Huang C.H. Huang	[39d883] initial revision
po	2019-03-21	C.H. Huang C.H. Huang	[b924b1] "Error Occurr
sounds	2012-11-04	C.H. Huang C.H. Huang	[39d883] initial revision
tests	2019-12-31	C.H. Huang C.H. Huang	[907768] update Copyri
uget	2019-12-31	C.H. Huang C.H. Huang	[907768] update Copyri
uglib	2020-02-27	C.H. Huang C.H. Huang	[6ee0ed] fix: ug_array_ir
ui-gtk	2024-01-15	faveoled faveoled	[7867bf] Use download
ui-gtk-1to2	2019-12-31	C.H. Huang C.H. Huang	[907768] update Copyri
.snapcraft.yaml	2017-04-28	Martin Wimpress Martin Wimpress	[a94222] Update snapcr
AUTHORS	2015-11-11	C.H. Huang C.H. Huang	[a49138] change versior

(a) Cropped segment of Sourceforge repository page with 18.29% damage due to missing commit author avatars, many duplicates for the same author



Cama has two class conditional components: a generator that transforms an image conditioned on a target class label , and a discriminator that predicts whether the low-level high-frequency pixel value dependency features of any given image conditioned on a label are real or fake. In addition, Cama has a fixed (w.r.t. its parameters) dual-stream discriminative decision-making component (evaluator) that decides whether a transformed image belongs to its target class. In essence, serves as a surrogate for the non-interactive black-box . W.r.t. , a transformed image is decomposed into its high and low spatial frequency components (and , respectively), via , with each assigned to a separate stream (and , respectively). The evaluator then reasons over the information present in and separately (via and , respectively). This reinforces the transformation process, as is constrained to transform both high and low spatial frequency camera model-specific artifacts used by the evaluator for discrimination.

During conditional adversarial training, minimizes:

whereas minimizes:

(b) Cropped segment of archived GitHub repository page measured at 49.2% damage caused by numerous missing full width and inline images scattered throughout repository README

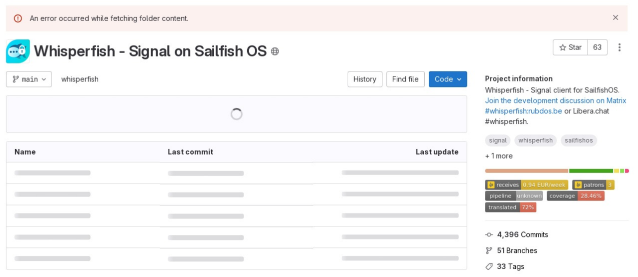
Figure 4: Highlights of missing image elements on two archived repository pages

stylesheets that cause elements to overlap and the page to become unreadable to viewers, as shown in Figure 4b.

Host	Sampled	Not Archived	Undamaged	Minor	Major
GH	10,000	5.24%	63.55%	29.78%	1.43%
GL	1,000	4.6%	54.5%	36.9%	4.0%
BB	610	90.1%	0.09%	0.002%	0%
SF	500	25.6%	18.8%	53.6%	2%
Total	12,110	10.33%	58.22%	29.85%	1.6%

Table 1: Damage for sampled repositories hosted at GitHub, GitLab, BitBucket, and Sourceforge

From the set of sampled repository pages, those hosted on GitHub were the most consistently available from the Wayback Machine. Numerous projects on Sourceforge and BitBucket had no mementos available and were only first archived due to the Memento Damage service triggering a crawl action in the Wayback Machine. Of note concerning our findings are those projects hosted through GitLab and BitBucket in which the vast majority are not able to be archived by the Internet Archive beginning around 2020. From what we can deduce, around this time GitLab began to use a more JavaScript-oriented approach that has caused issues with many crawlers, resulting in most projects after 2020 having only the general page skeleton and project overview information available but the "loading" placeholder element (gray rectangle) in place of the project's source table, depicted in Figure 5. Similarly, around the same time, it appears that BitBucket began to utilize a Web service called Sentry[17] where the page loads a minimal HTML skeleton and then checks are performed by the Sentry service before performing JavaScript fetch requests to retrieve and load the page information. Each of these affected pages contained an error related to an invalid "Data Source Name". For these cases, despite a lack of page content, the Memento Damage service was unable to detect page damage because no network requests were initiated to be measured for success or failure. Of the pages sampled from BitBucket, only the six archived before 2020 were able to be fully rendered on playback. Many home pages for Sourceforge projects were also originally missing from the Wayback Machine, with more than 100 (of 500) repositories not having been previously archived until a crawl from the Wayback Machine was triggered by the Memento Damage service upon inspection.



An error occurred while fetching folder content.

Whisperfish - Signal on Sailfish OS

Project information

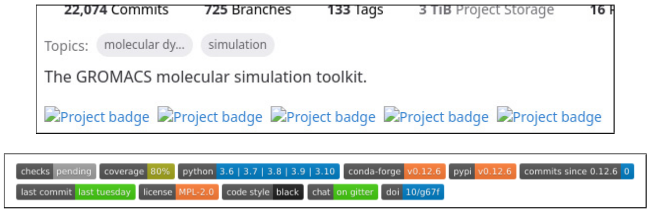
Whisperfish - Signal client for SailfishOS. Join the development discussion on Matrix #whisperfish:rubdos.be or Libera.chat #whisperfish.

4,396 Commits
51 Branches
33 Tags

Figure 5: Archived GitLab repository page for WhisperFish project. Only placeholder UI elements were captured instead of the true source table content that is loaded via JavaScript.

For most repository home pages, the largest visual element is the project's README. Much of the page content from the README and the surrounding page itself tended to be static text that is loaded as part of the initial page HTML DOM with damage mostly stemming from missing externally loaded page resources, such as images, embedded GIFs or videos, and stylesheet content. The primary contributors to page damage from missing content were missing user avatars, project logos in the page banner, and images for the dynamic shield badges as depicted in Figure 6. Shield badges are small images containing text against a colored background that are generated by an external service and then hot-linked into the project's README using a static URL and often used to show details about a project such as testing code coverage or release version information.

[17]https://sentry.io/


22,074 Commits 725 Branches 133 Tags 3 TiB Project Storage 16

Topics: molecular dy... simulation

The GROMACS molecular simulation toolkit.

Project badge Project badge Project badge Project badge Project badge

checks pending | coverage 80% | python 3.6 | 3.7 | 3.8 | 3.9 | 3.10 | conda-forge v0.12.6 | pypi v0.12.6 | commits since 0.12.6 0

last commit last tuesday | license MPL-2.0 | code style black | chat on gitter | doi 10/g67f

Figure 6: Example of missing and intact shield badges for Web hosted repository landing pages.

For the presentational pages of a repository, we find that, as they are largely comprised of text, they retain an overall usefulness for viewers seeking descriptive information about a repository. There might be significant details missing for various projects though, such as images and GIFs used in the project's README to display project screenshots for depicting various aspects or examples of project components.

5.2 Source Tree

In analyzing the archived source trees of the collected repositories hosted on GitHub, we found that there were typically only a small number of source files being archived overall. Around 65% (just over 6,000) did not have any source files archived. Of the remaining repositories that had archived source files, the majority had less than 5% those source file pages archived within the Wayback Machine, as shown in Figure 7, and three-quarters of the repositories had less than 15% archived. Ultimately, this highlights the lack a general reproducibility for archived software repositories present in the Wayback Machine in terms of being able to rebuild them using their archived source code from Web archives.

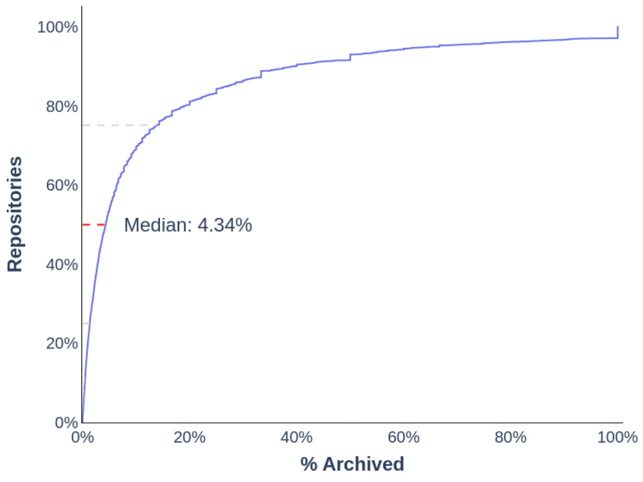
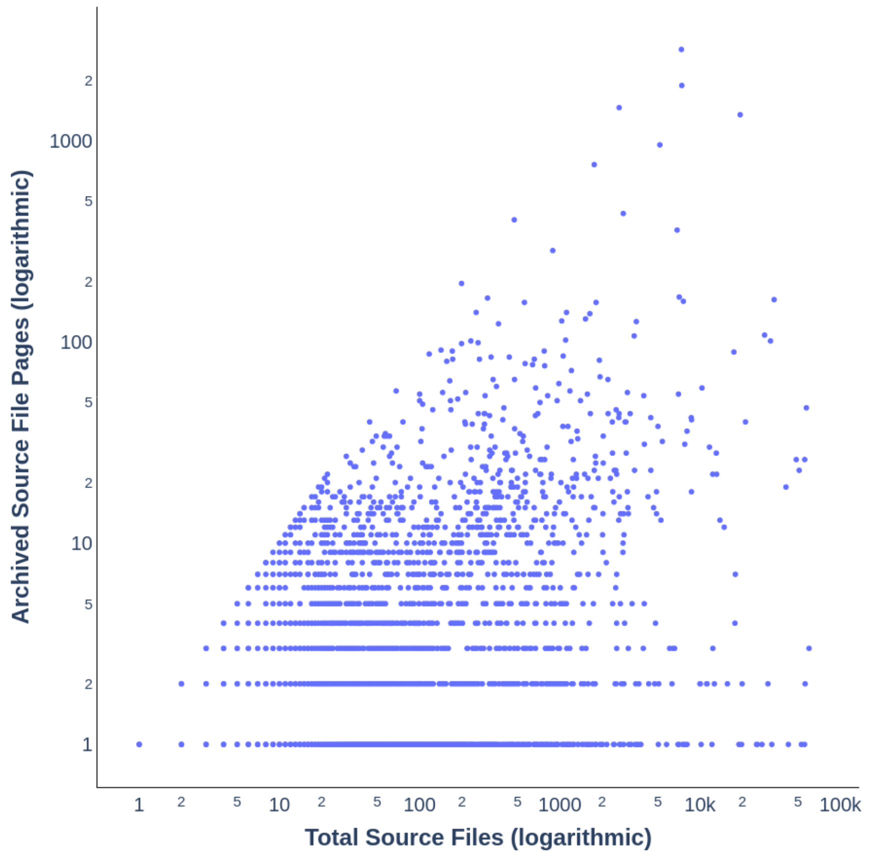


Figure 7: ECDF showing the percentage of repositories with at most x percentage of source files archived, for repositories with a non-zero amount of archived source files.

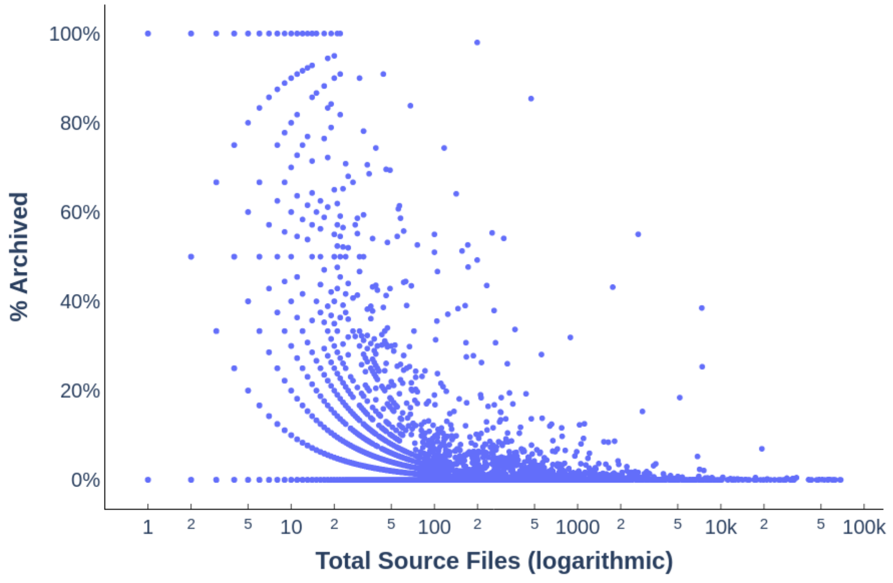
Due to the number of files and depth complexity of source trees, it would appear that more complex source trees generally had lower rates of availability in the Wayback Machine. Those repositories with higher rates of archival had most of their files at or near the source tree root. Only 94 repositories had their source trees completely archived, shown in Figure 8b, with the largest repository among these only containing a total of 22 files. Of those repositories completely archived, only 17 actually contained source code files, while the rest were composed of non-code files, such as Markdown, images, and PDF files, or used as container for a research paper, presentation, or list of Web URIs. When using Spearman's ρ and Kendall's τ_b metrics to correlate the archived percentage in relation to both file count and structural depth, both relations were found to have very weak correlations, despite the apparent archival trend. For archived percentage in relation to file count, there was a correlation of Spearman's $\rho = 0.246$ ($p = 3.325e$-128) and Kendall's $\tau_b = 0.191$ ($p = 2.825e$-128). In correlating repository archival with their structural depths, the correlations were Spearman's $\rho = 0.116$ ($p = 2.559e$-29) and Kendall's $\tau_b = 0.097$ ($p = 1.559e$-29). While these statistical correlations are low, we feel that the reasoning behind our findings lies partly in the behavioral mechanisms of the automated crawlers performing the archival of the various repositories. If a Web crawler is retrieving a list of trending repositories from GitHub and then visiting and indexing those pages, there seems to be a low likelihood that it is automatically visiting pages beyond that of the top level home pages gathered. This is indicated by the large number of home pages present from our GitHub sampling in comparison to the number of non-archived source file pages for each repository we examined. With an average file count among all analyzed GitHub repositories of around 505, it would not be impossible for an automated crawler to capture all of the Web pages representing source files, but unless explicitly programmed to recursively traverse all hyperlinks to a repository's source files, some source file pages will inevitably not be captured and the archived source tree would likely not have the ability to be reconstituted naturally.

Some outlier repositories can be seen (Figure 8b) with a high rate of archival and a not insignificant number of files. These include swift-evolution,[18] a documentation-oriented repository describing change proposals for the Apple programming language Swift, PowerShell,[19] a popular shell environment by Microsoft, and Konect-Toolbox,[20] a network analysis utility for MatLab. For Apple's swift-evolution repository, the bulk of the files are Markdown files linked not to the project's home page but one level lower, nested flatly inside the "proposals" directory. While these files might escape typical crawler behavior, this repository has over 10,000 stars, a measure of popularity akin to likes on social media posts, and are all linked to directly from the Swift's main documentation website.[21] This coupling of in-links from an external Web site, in conjunction with high user popularity driving traffic to these nested pages directly, is an example of how some repositories have higher rates of availability in the Wayback Machine not accounted for by statistical correlation. The PowerShell repository lags quite far behind that of swift-evolution with 55% of its source files archived, but it also has more than five times the number of total files in addition to a much more complex source tree structure.

[18]https://github.com/swiftlang/swift-evolution
[19]https://github.com/PowerShell/PowerShell
[20]https:/github.com/kunegis/konect-toolbox
[21]https://www.swift.org/swift-evolution/



(a) Number of source code Web pages archived vs. total files in the repository (log-log scale)



(b) Percentage of source code Web pages archived vs. total files in the repository

Figure 8: Archival of source code files in relation to the total number of files in each GitHub repository.

Although the PowerShell repository does not have numerous external links that help provide inroads directly to repository source files, it does have over four times the number of stars, demonstrating a level of popularity likely to improve its odds of being grabbed by a crawler. The Konect-Toolbox repository is an outlier, falling just shy of a full archival at 98% for 199 files. This repository is not too complex, having the majority of its files flatly in a subdirectory one level below its root [31]. The project's repository does not have a significant number of stars, though the project itself has more popularity across various social media websites and through its primary website that may be helping its higher archived percentage. The project's source files have also not changed much, with the latest update being only its README file two years ago. The most recent update behind that was four years ago, with further updates dating up to ten years ago. This largely static source tree and low rate of change are also likely beneficial in allowing more source file pages to become available in the Wayback Machine over time.

As Web crawling is a largely automated process and crawlers are not typically set up for depth-wise indexing, their tendency to grab only the content of the direct page they are crawling or content one hop away from it leads to a sharp drop-off in the amount of source files archived beyond the project's root source directory. Those projects that tended to fare the best in terms of the archival completeness of their source tree generally flatter directory structure and a smaller overall file count. This could explain much of the nested file pages of project source trees not being archived in the Wayback Machine, shown in Figure 9.

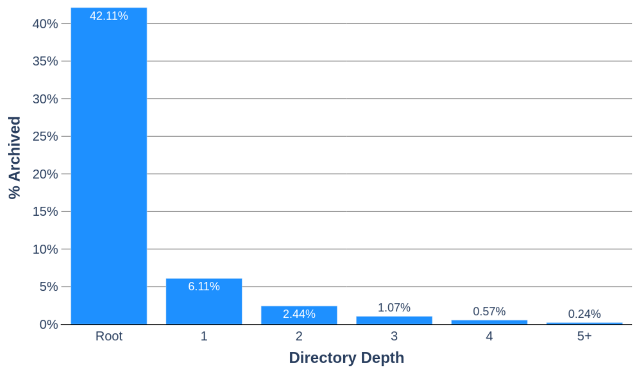
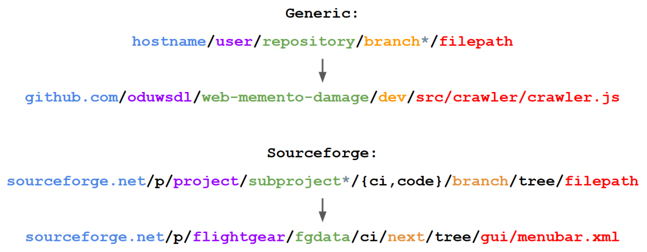


Figure 9: Percentage of source code file pages archived at various directory depths for Github projects with at least one source file archived.

Due to the root project directory being prominently displayed and its source files linked directly from a project's landing page, these source files had the highest archived percentage at just over 42% for repositories with a non-zero amount of archived source file availability. That rate sharply declined at each subsequent directory depth, being further and further hops away from the landing page, and dropped to just 6.11% for source directories one level deep, 2.44% at two levels, and then falling off to just over 1% and lower at subsequent levels. For our analysis, all directory depths beyond four were grouped, as there were a diminishing number of repositories utilizing a source tree structure with this level of complexity.

To note, the depth of a repository file or directory is not directly equivalent to the depth of its URI path. Most Web hosts typically use a pattern, shown in Figure 10, of hostname/user/repository/type/branch (optional if not default)/filepath, where the owner, repository name, type, and optional branch name form a prefixed path depth in the URI that is additional to the true depth of the source file or directory. Here, type is usually either "blob" for directories and "file" for files. In most cases, this constant of 3 or 4 can be added when mapping URI and source depths. For the Wayback Machine, this constant would also need to be increased to take into account the path prefixes added to the archived URI-R. Sourceforge is the exception to this rule, which utilizes a more complex schema of sourceforge.net/p/project/subproject/prefix/branch/type/filepath. Here, the subproject is optional, prefix is either "code" or "ci", and type is either "tree" or "blob". In Sourceforge's URI schema, there might not be an organization name, with the source code existing under a single path for the project name and optionally a subpath if subprojects existed. For our analysis and results, we exclusively measured the depth of the source files according to their filepath schema identifier from the URI schema, ignoring the beginning structure that differs among various repository hosting platforms.



```
Generic:
hostname/user/repository/branch*/filepath
↓
github.com/oduwsdl/web-memento-damage/dev/src/crawler/crawler.js

Sourceforge:
sourceforge.net/p/project/subproject*/{ci,code}/branch/tree/filepath
↓
sourceforge.net/p/flightgear/fgdata/ci/next/tree/gui/menubar.xml
```

Figure 10: URI schema example for Git hosting platforms. Asterisks indicate sections which may be omitted, such as the branch name in the case of the default branch name being used.

6 CONCLUSIONS AND FUTURE WORK

Our analysis of archived Web hosted software repositories highlights the shortcomings of traditional archival methodologies and hosts to faithfully capture the depth of content required to rebuild such projects. We used a combination of API calls for the Internet Archive's Wayback Machine and page crawls using the Memento Damage tool to carry out page analysis for over 10,000 archived repository URIs from GitHub and around 2,000 secondarily from GitLab, BitBucket, and Sourceforge. With regard to the presentational components of hosted repositories, we found that more than 10% had not been archived and over 30% had some form of page damage with respect to archived page fidelity. The recent use of JavaScript for the loading of page content on both GitLab and BitBucket also prevented numerous repository pages from being available or usable within the Wayback Machine. An average user without any knowledge of Web archival might see a broken project home page and assume that the entire project is missing. Even if they were knowledgeable about advanced search techniques, such as using the Wayback Machine's CDX API, should there be archived source files for such a project, these pages are likely to suffer the same fate as its home page and be unusable for a viewer. For a user seeking general information about an archived repository on platforms not utilizing JavaScript for page loading, they are most likely to be able to view the text information contained in a project's README when rendered as part of its home page but might not be able to gain useful information should key images be missing from the archived page, such as examples or screenshots. We also measured the extent of source code archival across the collected GitHub repositories. We found that the vast majority of archived repositories could not be rebuilt exclusively from their archived source content within the Wayback Machine. Of the repositories analyzed, an average of 4.72% of source files were archived. This low average is in part due to only around 35% of analyzed projects having their source content archived at all. Of those repositories with a non-zero number of archived files, there was still only 13.39% of each repository archived on average. These source pages are also subject to potential issues due to the usage of JavaScript for loading page content, though the more prevalent issue with the archival of source trees lies in their structure. As each source file is presented using individual HTML pages, archival crawlers must visit each one to completely capture the content of a source tree or risk the archive not having the ability to be reconstructed. As this level of depth is not standard for crawlers archiving the Web, many of the Web pages for deeply nested source code files, which can often be the most crucial or specialized, are not crawled or archived.

With respect to our analysis of project source trees, we chose to focus on the completeness of archived content. One important factor we did not address was that of temporal coherence with regard to the source files themselves. As software projects have different release versions or milestones over time, the consistency of source file content can change and the archival date becomes a more important factor in determining source validity. Future research will incorporate such temporal analysis as improvements are made to better assess the validity of archived source files for specific software release versions by performing rigorous analysis of changes to the timestamps and commit history of a given software repository. Further improvements to our analysis might be made through static analysis of the code base to improve analytical efficiency and determine source files that are not utilized and could be excluded from the analysis.

Looking to the future, avenues for further research that could be explored concerning this topic include the expansion of analysis to the complete body of descriptive or presentational pages for repositories, such as wiki, issue, and testing pages. Another interesting avenue of research is the potential to use artificial intelligence systems to fill in the gaps of missing archived source files through code synthesis. For this, machine learning techniques could be harnessed in conjunction with the context provided by other available source files and project information to attempt to generate code files that would enable an incompletely archived repository to be successfully recompiled.

ACKNOWLEDGMENTS

This work is supported in part by a grant from the Alfred P. Sloan Foundation. We thank our many colleagues at the Web Science and Digital Libraries research group at Old Dominion University, NYU Division of Libraries, and the Research Library at Los Alamos National Laboratory.

REFERENCES

[1] Jean-François Abramatic, Roberto Di Cosmo, and Stefano Zacchiroli. 2018. Building the universal archive of source code. *Commun. ACM* 61, 10 (2018), 29–31.

[2] Vangelis Banos and Yannis Manolopoulos. 2016. A quantitative approach to evaluate Website Archivability using the CLEAR+ method. *International Journal on Digital Libraries* 17, 2 (2016), 119–141. https://doi.org/10.1007/s00799-015-0144-4

[3] Avijit Bhattacharjee, Sristy Sumana Nath, Shurui Zhou, Debasish Chakroborti, Banani Roy, Chanchal K. Roy, and Kevin Schneider. 2020. An Exploratory Study to Find Motives Behind Cross-platform Forks from Software Heritage Dataset. In *Proceedings of the 17th International Conference on Mining Software Repositories* (Seoul, Republic of Korea) *(MSR '20)*. Association for Computing Machinery, New York, NY, USA, 11–15. https://doi.org/10.1145/3379597.3387512

[4] Haley Bragg, Himarsha Jayanetti, Michael L. Nelson, and Michele C. Weigle. 2023. Less than 4% of Archived Instagram Account Pages for the Disinformation Dozen are Replayable. In *Proceedings of ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. 102–106. https://doi.org/10.1109/JCDL57899.2023.00025

[5] Justin Brunelle. 2016. *Scripts in a Frame: A Framework for Archiving Deferred Representations.* PhD thesis. Old Dominion University, Norfolk, VA. https://doi.org/10.25777/k8px-z178

[6] Justin Brunelle, Mat Kelly, Hany SalahEldeen, Michele C. Weigle, and Michael L. Nelson. 2015. Not All Mementos Are Created Equal: Measuring the Impact of Missing Resources. *International Journal of Digital Libraries (IJDL)* 16, 3-4 (September 2015), 283–301. https://doi.org/10.1007/s00799-015-0150-6

[7] Justin F. Brunelle, Michele C. Weigle, and Michael L. Nelson. 2017. Archival Crawlers and JavaScript: Discover More Stuff but Crawl More Slowly. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. 1–10. https://doi.org/10.1109/JCDL.2017.7991554

[8] Garret Christensen, Allan Dafoe, Edward Miguel, Don A. Moore, and Andrew K. Rose. 2019. A study of the impact of data sharing on article citations using journal policies as a natural experiment. *PLOS ONE* 14, 12 (12 2019), 1–13. https://doi.org/10.1371/journal.pone.0225883

[9] Ludovic Courtès, Timothy Sample, Simon Tournier, and Stefano Zacchiroli. 2024. Source Code Archiving to the Rescue of Reproducible Deployment. In *Proceedings of the ACM Conference on Reproducibility and Replicability*. ACM, 1–10. https://doi.org/10.1145/3641525.3663622

[10] Roberto Di Cosmo. 2020. Archiving and Referencing Source Code with Software Heritage. In *Proceedings of the International Congress on Mathematical Software (ICMS)*. Springer International Publishing, 362–373.

[11] Roberto Di Cosmo, Morane Gruenpeter, and Stefano Zacchiroli. 2018. Identifiers for Digital Objects: the Case of Software Source Code Preservation. In *Proceedings of the 15th International Conference on Digital Preservation (iPRES)*. Boston, United States, 1–9. https://doi.org/10.17605/OSF.IO/KDE56

[12] Roberto Di Cosmo, Morane Gruenpeter, and Stefano Zacchiroli. 2020. Referencing Source Code Artifacts: A Separate Concern in Software Citation. *Computing in Science & Engineering* 22, 02 (2020), 33–43. https://doi.org/10.1109/MCSE.2019.2963148

[13] Roberto Di Cosmo and Stefano Zacchiroli. 2017. Software Heritage: Why and How to Preserve Software Source Code. In *Proceedings of the 14th International Conference on Digital Preservation (iPRES)*. Kyoto, Japan, 1–10. https://hal.science/hal-01590958

[14] Roberto Di Cosmo and Stefano Zacchiroli. 2023. The Software Heritage Open Science Ecosystem. In *Software Ecosystems: Tooling and Analytics*, Tom Mens, Coen De Roover, and Anthony Cleve (Eds.). Springer International Publishing, Cham, 33–61. https://doi.org/10.1007/978-3-031-36060-2_2

[15] Chris DiBona. 2015. Bidding farewell to Google Code. https://opensource.googleblog.com/2015/03/farewell-to-google-code.html

[16] Florian Echtler and Maximilian Häußler. 2018. Open Source, Open Science, and the Replication Crisis in HCI. In *Proceedings of the Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI EA '18)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3170427.3188395

[17] Emily Escamilla, Talya Cooper, Vicky Rampin, Martin Klein, Michele C. Weigle, and Michael L. Nelson. 2022. The Rise of GitHub in Scholarly Publications. In *Proceedings of the Theory and Practice of Digital Libraries Conference (TPDL)*. https://doi.org/10.1007/978-3-031-16802-4_15

[18] Emily Escamilla, Martin Klein, Talya Cooper, Vicky Rampin, Michele C. Weigle, and Michael L. Nelson. 2023. Cited But Not Archived: Analyzing the Status of Code References in Scholarly Articles. In *Proceedings of the International Conference on Asia-Pacific Digital Libraries (ICADL)*. https://doi.org/10.1007/978-981-99-8088-8_17

[19] Emily Escamilla, Lamia Salsabil, Martin Klein, Jian Wu, Michele C. Weigle, and Michael L. Nelson. 2023. It's Not Just GitHub: Identifying Data and Software Sources Included in Publications. In *Proceedings of the Theory and Practice of Digital Libraries Conference (TPDL)*. 195–206. https://doi.org/10.1007/978-3-031-43849-3_17

[20] Greg Foster. 2024. How GitHub monopolized code hosting. https://graphite.dev/blog/github-monopoly-on-code-hosting

[21] Kritika Garg, Himarsha R Jayanetti, Sawood Alam, Michele C Weigle, and Michael L Nelson. 2024. Challenges in replaying archived Twitter pages. *International Journal on Digital Libraries* 25, 2 (2024), 217–236.

[22] Wilhelm Hasselbring, Leslie Carr, Simon Hettrick, Heather S. Packer, and Thanassis Tiropanis. 2019. *FAIR and Open Computer Science Research Software.* Technical Report arXiv:1908.05986. arXiv. http://arxiv.org/abs/1908.05986

[23] Friederike Hendriks, Dorothe Kienhues, and Rainer Bromme. 2020. Replication crisis = trust crisis? The effect of successful vs failed replications on laypeople's trust in researchers and research. *Public Understanding of Science* 29, 3 (2020), 270–288. https://doi.org/10.1177/0963662520902383 PMID: 32036741.

[24] Simon Hettrick. 2014. It's impossible to conduct research without software, say 7 out of 10 UK researchers. https://www.software.ac.uk/blog/its-impossible-conduct-research-without-software-say-7-out-10-uk-researchers

[25] Shawn M. Jones, Herbert Van de Sompel, Harihar Shankar, Martin Klein, Richard Tobin, and Claire Grover. 2016. Scholarly Context Adrift: Three out of Four URI References Lead to Changed Content. *PLOS ONE* 11, 12 (12 2016), 1–32. https://doi.org/10.1371/journal.pone.0167475

[26] Mat Kelly, Michael L. Nelson, and Michele C. Weigle. 2014. The Archival Acid Test: Evaluating Archive Performance on Advanced HTML and JavaScript. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. London, 25–28. https://doi.org/10.1109/JCDL.2014.6970146

[27] Johannes Kiesel, Florian Kneist, Milad Alshomary, Benno Stein, Matthias Hagen, and Martin Potthast. 2018. Reproducible Web Corpora: Interactive Archiving with Automatic Quality Assessment. *Journal of Data and Information Quality* 10, 4, Article 17 (Oct 2018), 25 pages. https://doi.org/10.1145/3239574

[28] Martin Klein, Herbert Van de Sompel, Robert Sanderson, Harihar Shankar, Lyudmila Balakireva, Ke Zhou, and Richard Tobin. 2014. Scholarly Context Not Found: One in Five Articles Suffers from Reference Rot. *PLOS ONE* 9, 12 (12 2014), 1–39. https://doi.org/10.1371/journal.pone.0115253

[29] Petros Maniatis, Mema Roussopoulos, T. J. Giuli, David S. H. Rosenthal, and Mary Baker. 2005. The LOCKSS peer-to-peer digital preservation system. *ACM Transactions on Computer Systems* 23, 1 (Feb. 2005), 2–50. https://doi.org/10.1145/1047915.1047917

[30] Vadim Markovtsev and Waren Long. 2018. Public Git archive: A big code dataset for all. In *Proceedings of the 15th International Conference on Mining Software Repositories* (Gothenburg, Sweden) *(MSR '18)*. Association for Computing Machinery, New York, NY, USA, 34–37. https://doi.org/10.1145/3196398.3196464

[31] Corren McCoy. 2017. Pagination Considered Harmful to Archiving. https://ws-dl.blogspot.com/2017/09/2017-09-13-pagination-considered.html

[32] Genevieve Milliken. 2021. IASGE Environmental Scan. https://osf.io/ku24q/.

[33] Sarah Nguyen. 2021. Gap Analysis of the Scholarly Git Experience. https://osf.io/q27m9/.

[34] Sarah Nguyen and Vicky Rampin. 2022. Who Writes Scholarly Code?. In *Proceedings of International Digital Curation Conference (IDCC)*. https://doi.org/10.5281/zenodo.6670225

[35] Antoine Pietri. 2021. *Organizing the graph of public software development for large-scale mining.* Theses. Université Paris Cité. https://hal.science/tel-03515795

[36] Antoine Pietri, Diomidis Spinellis, and Stefano Zacchiroli. 2020. The Software Heritage Graph Dataset: Large-scale Analysis of Public Software Development History. In *Proceedings of the 17th International Conference on Mining Software Repositories* (Seoul, Republic of Korea) *(MSR '20)*. Association for Computing Machinery, New York, NY, USA, 1–5. https://doi.org/10.1145/3379597.3387510

[37] Victoria Reich. 2008. CLOCKSS—It Takes a Community. *The Serials Librarian* 54, 1-2 (2008), 135–139. https://doi.org/10.1080/03615260801973968

[38] Lamia Salsabil, Jian Wu, Muntabir Hasan Choudhury, William A. Ingram, Edward A. Fox, Sarah M. Rajtmajer, and C. Lee Giles. 2022. A Study of Computational Reproducibility using URLs Linking to Open Access Datasets and Software. In *Companion Proceedings of the Web Conference* (Virtual Event, Lyon, France) *(WWW '22)*. Association for Computing Machinery, New York, NY, USA, 784–788. https://doi.org/10.1145/3487553.3524658

[39] Franklin Sayre and Amy Riegelman. 2018. The Reproducibility Crisis and Academic Libraries. *College & Research Libraries* 79, 1 (2018), 2. https://doi.org/10.5860/crl.79.1.2

[40] Len Shustek. 2006. What Should We Collect to Preserve the History of Software? *IEEE Annals of the History of Computing* 28, 4 (2006), 112–111. https://doi.org/10.1109/MAHC.2006.78

[41] Erika Siregar. 2017. Deploying the Memento-Damage Service. https://ws-dl.blogspot.com/2017/11/2017-11-22-deploying-memento-damage.html

[42] Vicky Steeves, Genevieve Milliken, and Sarah Nguyen. 2020. Let's Talk Git! Investigating and Archiving the Scholarly Git Experience. In *Proceedings of code4Lib 2020*. Pittsburgh, Pennsylvania, USA.

[43] Vicky Steeves, Genevieve Milliken, Sarah Nguyen, Wendy Hagenmaier, Euan Cochrane, Alexandra Chassanoff, and Morane Gruenpeter. 2020. Saving Software for Future [Re]use. In *Proceedings of Archives*Records, the Annual Meeting of the Society of American Archivists (SAA)*.

[44] Vicky Steeves and Sarah Nguyen. 2020. commit-ment issues with Git: investigating & archiving y'alls work. In *Proceedings of csv,conf,v5*.

[45] Herbert Van de Sompel, Michael Nelson, and Robert Sanderson. 2013. HTTP Framework for Time-Based Access to Resource States – Memento. RFC 7089. https://doi.org/10.17487/RFC7089

[46] Supatsara Wattanakriengkrai, Bodin Chinthanet, Hideaki Hata, Raula Gaikovina Kula, Christoph Treude, Jin Guo, and Kenichi Matsumoto. 2022. GitHub repositories with links to academic papers: Public access, traceability, and evolution. *Journal of Systems and Software* 183 (2022), 111117. https://doi.org/10.1016/j.jss.2021.111117

[47] Michele C. Weigle, Michael L. Nelson, Sawood Alam, and Mark Graham. 2023. Right HTML, Wrong JSON: Challenges in Replaying Archived Webpages Built with Client-Side Rendering. In *Proceedings of ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. 82–92. https://doi.org/10.1109/JCDL57899.2023.00022