GHAgentFiles: A dataset of coding agent file histories in GitHub repositories

Guillaume Cardoen
Software Engineering Lab
University of Mons
Mons, Belgium
guillaume.cardoen@umons.ac.be

Tom Mens
Software Engineering Lab
University of Mons
Mons, Belgium
tom.mens@umons.ac.be

Alexandre Decan
F.R.S.-FNRS Research Associate
Software Engineering Lab
University of Mons
Mons, Belgium
alexandre.decan@umons.ac.be

***Abstract*—Since the introduction of coding-specific LLMs in the early 2020s, coding agents have become increasingly popular in collaborative software development projects. Since 2025, it is common practice to specify context files that provide additional context to coding agents about the project structure, development commands, coding conventions, and so on. To enable large-scale empirical studies of collaborative GitHub repositories involving coding agents, we provide a dataset of context, skill and subagent files and an open-source tool to identify and extract such files automatically. Out of an initial list of 165K+ repositories, we identify 27K+ repositories with coding agent files for six major agents (Claude, Codex, GitHub Copilot, Gemini, Windsurf and Cursor), resulting in 142K+ distinct file histories and 401K+ revisions covering their evolution. We conduct a preliminary observational analysis on this large dataset and present some perspectives to the research community on how to use this dataset for future empirical studies.**

***Index Terms*—dataset, coding agent, context file, collaborative software development, empirical software engineering**

I. Introduction

Since the public release of ChatGPT in 2022, large language models (LLMs) have become widely used in a plethora of application domains. In the area of collaborative software development specifically, coding-specific LLMs have become commonplace. Since 2025, (semi-)autonomous coding agents have started to automate a wide range of software development activities [1]. In an ideal scenario, a software engineer would *prompt* the agent in natural language to perform a specific task, and the agent would autonomously carry it out by making the necessary changes to the software project.

To increase their autonomy and performance, agents can benefit from *context files*, providing contextual information and guidance on the project's conventions, architecture and style guidelines, building, testing and deployment instructions, how to maintain quality and security, agent boundaries, and so on.[1] These *context files* tend to be expressed in natural language and stored in *Markdown format* at specific locations in the project's repository. They are meant to be read by coding agents that will use them as context to accomplish their assigned tasks. Numerous examples of context files are available online.[2]

[1]https://github.blog/ai-and-ml/github-copilot/how-to-write-a-great-agents-md-lessons-from-over-2500-repositories/

[2]See, for example, https://agents.md

Given the increasing popularity of AI agents, this paper contributes a large dataset of context files histories, to enable replication of existing studies on a larger dataset, as well as to conduct new empirical studies at scale. To the best of the authors' knowledge, at the time of writing, this is the largest dataset of context files covering the largest period of time. The dataset contains 142K+ file histories from 27K+ repositories, covering 401K+ revisions until 1 July 2026.

II. Related Work

Prior empirical studies have focused on the structure, maintenance, and adoption of context files, as well as their impact on agents [1], [2], [3], [4]. Some recent studies have examined context files using limited datasets. Mohsenimofidi *et al.* [5] investigated the adoption of context files in open-source software (OSS) repositories, along with the content and evolution of those files. They constructed a dataset of context files from 453 repositories, focusing on CLAUDE.md, AGENTS.md, and GEMINI.md. They found that 5% of the repositories used context files, primarily to provide context about conventions, guidelines, project architecture, and structure. The most common changes to these files involved adding instructions, while modifying instructions was rare.

Chatlatanagulchai *et al.* [6] created a dataset of 2.3K+ context files from 1.9K+ repositories, focusing on CLAUDE.md, AGENTS.md, and copilot-instructions.md. They found that context files follow a consistent hierarchy with markdown headers, but tend to be long and difficult to read. These files are heavily maintained and primarily provide actionable guidance for executing and maintaining code.

Galster *et al.* [7] extracted 4,768 context files from 2,853 repositories, focusing on Claude, Copilot, Codex, Cursor, and Gemini. They found that context files serve as the primary configuration mechanism for coding agents, with AGENTS.md files emerging as an open standard. In a follow-up work [8], they extended their dataset to 4,738 repositories.

Given the recent nature of context files, all these studies relied on datasets that were quite limited in size and covering a short period of time. This paper contributes by providing a substantially larger dataset of 142,294 context file histories and their metadata, representing a total of 401,873 revisions, obtained from 27,717 GitHub repositories. We also contribute

an open-source tool that we developed to create the dataset and that can be used to further expand it. By doing so, we provide the software community with the necessary means to enable large empirical studies on the evolution of context files for coding agents in GitHub, thereby alleviating the error-prone and time-consuming nature of producing such datasets.

III. Data Collection

To identify and extract context files from GitHub repositories, we developed cofee (an acronym for COntext File Extraction Engine), a CLI-based open-source tool. It extracts file histories from a (local or online) git repository by issuing the command cofee -c config.toml repo, where repo is the repository name and config.toml declares the files to extract. Valid paths are specified in the .toml configuration file via a simple glob-like syntax, exemplified in Listing 1. The example listing instructs to extract and classify any file named AGENTS.md into the generic group. Files named CLAUDE.md and variations are classified in the claude group, and similar for the other groups.

Listing 1: Illustration of the configuration file syntax. The **/ pattern signifies that we are searching for files in any directory or subdirectory of the repository.

```
[filters]
generic = ['**/AGENTS.md']
claude =  ['**/CLAUDE.md',
           '**/CLAUDE.local.md',
           '**/.claude/rules/*.md']
cursor = ['**/.cursor/rules/*.md',
          '**/.cursor/rules/*.mdc',
          '**/.cursorrules']
gemini =  ['**/GEMINI.md',
           '**/.gemini/system.md']
copilot = [
    '**/.github/copilot-instructions.md',
    '**/.github/instructions/*.instructions.md',
    '**/.github/instructions/**/*.instructions.md']
...
```

For a given repository, cofee iterates through all ancestor commits of a given commit reference, defaulting to git HEAD, and stops either once all ancestors are considered, or at a specified git commit reference. We follow the first-parent rule, stating that only the first parent of each commit is followed when traversing the git history. In the case of a merge commit, we thus follow only one branch. At each commit, cofee considers every touched file, and checks whether it should be extracted based on its path. If the file path matches one of the specified paths, the corresponding file is stored locally alongside its git metadata (such as the commit author and the committed date). Unless specified otherwise by an optional argument, the metadata is sent to the standard output.

We focused our attention on six coding agents (Claude, Codex, GitHub Copilot, Gemini, Windsurf and Cursor), corresponding to those reported as being the most prevalent in previous studies [1], [4], [7]. The .toml configuration file that we used to obtain our dataset of context file histories can be found in our replication package (see Section VII). This configuration file was obtained through the following process.

Each author independently inspected the online documentation of the six coding agents, with the aim to identify possible locations and naming conventions of context files for those agents. They then jointly discussed in order to resolve ambiguous cases and come to an agreement. They came up with seven groups, one for each agent plus a generic group corresponding to the emerging AGENTS.md standard that can be used by most of the considered agents.

In addition to *context files*, we also extracted *skill files* that can be regarded as reusable workflow contexts describing how to do a specific task following a sequence of steps, and *subagent files* that can be seen as context files defining the context of a specialised agent to which an orchestrating agent will delegate specific tasks. The patterns for detecting these files can be found in our replication package.

To obtain a list of candidate GitHub repositories for extracting context files, we queried the SEART GitHub search engine [9] on 1 July 2026. The motivation of using SEART is that it allows to filter relevant repositories independent of their usage of agents, hence also enabling the quantification of agent adoption rate in GitHub. We focused on popular and recently active repositories, excluding personal and experimental repositories [10], by applying the following filters to each repository: (a) created at least a year ago; (b) not being a fork; (c) having at least 300 commits; (d) having at least one commit in the last year.

This resulted in a list of 165,281 candidate repositories, that were cloned locally between 1 July 2026 and 2 July 2026. Using cofee, we traversed the git history of each of these repositories to extract all context, skill, and subagent files. Through manual inspection of the extracted files, we noticed that some files, corresponding to correct matches according to our configuration, did not correspond to actual context files intended to be read by any agent. In particular, we found examples of files matching context file names that were created well before the introduction of LLMs and coding agents. We took November 2022, the official introduction date of ChatGPT,[3] as a proxy for the earliest possible introduction date of context files. Files with an earlier creation date are extremely unlikely to correspond to context files, since LLMs did not yet exist before that period, and the notion of context files was only introduced several years later. Similarly, we found files matching skill file names well before October 2025, the official introduction date of skills in Claude,[4] the earliest tool known to support skill files. We therefore ignored any history of context files created before November 2022 and skill files created before October 2025, excluding 136 disctinct file histories corresponding to 719 file revisions.

To validate the dataset, we selected a random sample of 666 files, corresponding to a 5% margin of error with a 99% level of confidence. All three authors independently reviewed these files to answer the question "*Is this a context file intended to be read by an agent?*" with three possible responses:

[3]https://openai.com/index/chatgpt/

[4]https://claude.com/blog/skills

TABLE I: Metadata stored for each file revision

field	description
repository	The repository from which the context file was extracted.
agent_name	The agent group (*e.g.*, claude) to which the file belongs.
category	The file category: context, skill or subagent.
uid	A unique file identifier surviving content modifications and file renamings.
commit_hash	The git commit hash.
author_name	The author that touched this file.
author_email	The email of that author.
committer_name	The committer name.
committer_email	The email of that committer.
commit_date	The commit date.
authored_date	The authored date of the commit.
file_path	The path to the file in the repository.
previous_file_path	The path to the file before it was touched.
file_hash	The hash of the file contents.
previous_file_hash	The hash of the file contents before the file touch.
git_change_type	The type of change made to the file, as provided by git (*i.e.*, **A**dded, **D**eleted, **M**odified or **R**enamed).
symbolic_link	A boolean flag signaling whether the file is a symbolic link (*i.e.*, a pointer or alias to another file).
previous_symbolic_link	A boolean flag signaling whether the file was a symbolic link before it was touched.

"*Yes*", "*No*" or "*Maybe*". The third response corresponds to cases where a file might not only serve as a context file, but might also be used for other purposes (*e.g.*, human contributor guidelines). All three authors unanimously responded "*Yes*" for 509 files (76.4%) and "*Maybe*" for 106 files (15.9%). Two authors answered "*Yes*" for the remaining 51 files (7.7%), while one author answered "*Maybe*" for those. This resulted in an interrater agreement of 92.4%. The validation process confirmed the dataset's quality, since no one responded "*No*" for any of the files.

IV. Dataset of Context File Histories

Using the process described in Section III, we created GHAgentFiles, a large dataset of context, skill and subagent file histories, containing metadata for each file revision (presented in Table I). We extracted 142,294 file histories from 27,717 distinct GitHub repositories. The earliest commit touching a file is 22 November 2022, while the latest is 1 July 2026. The dataset also includes 1,086 symbolic links, *i.e.*, pointers or aliases referencing another file. Unlike regular files, symbolic links are not modified when their target reference is. This implies that empirical analyses using the dataset should treat symbolic links separately. For this reason, they are explicitly flagged as such in the dataset.

The dataset is stored in multiple compressed files, as described in Table II. To prevent name collisions and reduce the overall dataset size, each file revision is stored under a filename corresponding to the SHA-256 hash of its contents.

Table III presents the dataset characteristics. A total of 401,873 file revisions are contained in a total of 185,795 commits. In the latest snapshot (*i.e.*, the last commit of each repository), the dataset contains a total of 111,727 file histories

TABLE II: Content of the dataset

file	description
all_content.tar.gz	Compressed archive containing all extracted files, including context, skills and subagents.
context_content.tar.gz	Compressed archive containing only extracted context files, ignoring skills and subagents.
all_metadata.csv.gz	Compressed CSV file containing metadata of all extracted files, such as committer name, email and commit date (see Table I).
context_metadata.csv.gz	Compressed CSV file containing only metadata for the context files, ignoring skills and subagents.
repositories.csv.gz	Compressed CSV file obtained from SEART [9], containing metadata of each considered repository (*e.g.*, number of stars, contributors, main language, ...)

TABLE III: Characteristics of the GHAgentFiles dataset. The % indicates the proportion of elements in the last snapshot w.r.t. all snapshots.

characteristic	all snapshots	last snapshot
# repositories	27,717	26,577 (95.9%)
with context files	26,019	24,836 (95.5%)
with skill files	6,919	6,564 (94.9%)
with subagent files	1,615	1,348 (83.5%)
# histories	142,294	111,727 (78.5%)
for context files	72,100	58,372 (81.0%)
for skill files	62,055	47,572 (76.7%)
for subagent files	8,350	5,783 (69.3%)
# commits	185,795	50,783 (27.3%)
touching context files	150,521	36,956 (24.6%)
touching skill files	41,004	14,930 (36.4%)
touching subagent files	5,402	2,018 (37.4%)
# file revisions	401,873	111,727 (27.8%)
touching context files	239,240	58,372 (24.4%)
touching skill files	145,050	47,572 (32.8%)
touching subagent files	17,583	5,783 (32.9%)

(and the same number of file revisions since we consider a single snapshot) from 50,783 commits. This corresponds to 27.3% of all commits and 27.8% of all file revisions in the entire dataset. Commits in this snapshot contain an average of 4.20 context/agent/subagent files with a median of 1, indicating a skewed distribution.

The heatmap of Figure 1 quantifies how file revisions in the

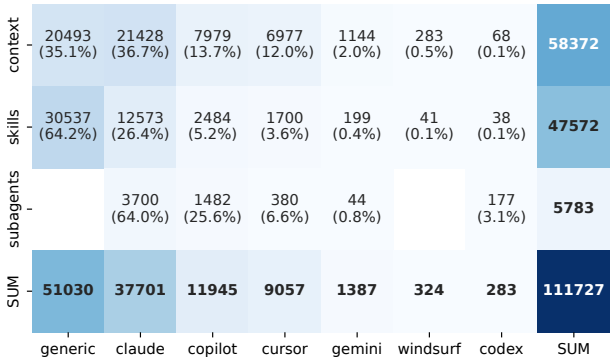

Fig. 1: Heatmap quantifying the files belonging to each agent group and file category in the latest snapshot. Empty cells indicate cases for which no data is available.

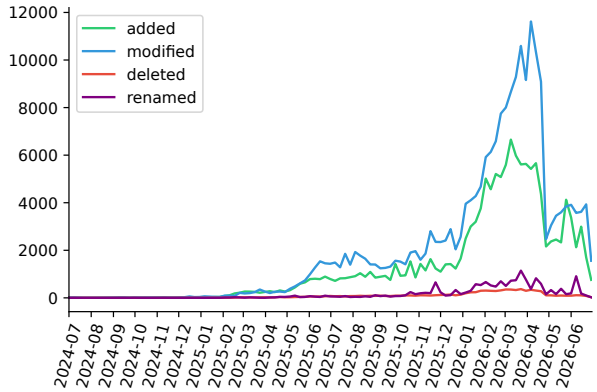

Fig. 2: Number of files touched by commits on a weekly basis (based on the commit date), grouped by change type (added, deleted, modified and renamed).

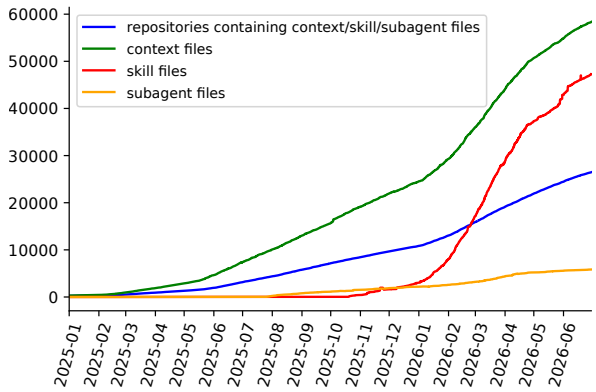

Fig. 3: Evolution of the number of context, skill and subagent files and their containing repositories. Dashed lines represent exponential regression models fit to the data.

latest snapshot are spread over agent groups and file categories. In terms of file categories, 52.2% of all file revisions are context files, 42.6% skill files, and 5.2% subagent files. In terms of the seven agent groups, 45.7% of all file revisions correspond to the generic context file AGENTS.md, while 33.7% belong to Claude. Copilot (10.7%) and Cursor (8.11%) are less prevalent. The three remaining coding agents are infrequently present, possibly because those agents are using file name conventions captured by the generic group.

Narrowing down to the context file category, we observe that Claude (36.7%) and the generic AGENTS.md files (35.1%) take the lion share with similar proportions. For skill files, the generic SKILL.md corresponds to a large majority of 64.2%, while SKILL.md in a .claude directory corresponds to 26.4%. For subagents files, a majority (64.0%) corresponds to Claude subagents, and more than a quarter (25.6%) corresponds to Copilot subagents.

V. Exploratory Observational Analysis

The proposed dataset can be used to conduct empirical studies of coding agent files, such as those initiated in [5], [6], [7]. This section presents an exploratory evolutionary analysis of the adoption of context files by GitHub repositories.

As a first goal, we aimed to understand how agent files are evolving over the observation period covered by the dataset. Figure 2 visualises the weekly number of files touched by commits (*i.e.*, added, modified, deleted or renamed). One can observe that coding agent files start to emerge roughly since early 2025. For early 2026, we observe a sharp peak in the number of files being added and modified. As one would expect, more files being modified than new ones being added. Removal or renaming of files remains infrequent.

In the subsequent analyses, we focus on the period starting from January 2025, since there is little data prior to that point in time. Figure 3 breaks down the growth in number of files per file category (*i.e.*, context, skill, and subagent files). Focusing on context files, their number is slowly growing until June 2025, after which the slope of the growth curve increases. In December 2025, another phase with an even steeper slope

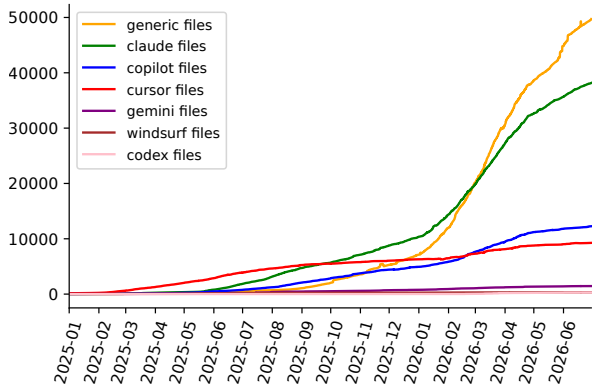

Fig. 4: Evolution of files grouped by coding agent.

begins. After April 2026, the growth slightly decreases. Skill files seem to take off in November 2025, with an inflection point in March 2026, where the growth is declining again. Subagents are less used overall, starting around August 2025 with a smaller number and lower slope than context files and skill files.

Figure 4 breaks down the analysis per agent group. The "generic" group has become the most popular, reflecting that AGENTS.md is becoming a standard naming practice across all major coding agents. Claude has taken over Cursor in popularity around October 2025 and has stayed the most important non-generic group ever since. Copilot is slowly increasing momentum, probably due to its tight integration into Github, becoming the third most important group at the end of the observation period, though staying far below Claude and the generic group.

During our manual validation (*c.f.* Section III), we observed that many files were structured with Markdown headings. Therefore, we explored the headings titles within the context files of the latest snapshot. We only extracted second-level headings (present in 86.6% of all context files) and counted their frequency. We excluded first-level headings, as they were found to be predominantly used as file titles (*e.g.*, *"# AGENTS.md"*) rather than actually reflecting a specific

TABLE IV: Top 10 of the most frequent case-insensitive second-level headings in context files.

heading	count	proportion of context files
project overview	7,442	16.7%
architecture	6,841	15.3%
testing	4,423	9.9%
commands	2,997	6.7%
overview	2,759	6.2%
project structure	2,620	5.9%
code style	2,144	4.8%
development commands	1,982	4.4%
development workflow	1,712	3.8%
conventions	1,657	3.7%

structure within the file. Table IV presents the 10 most frequent headings in context files. Such headings seem to give an overall overview of the project structure, or describe the architecture of the project. Other common subsections relate to testing, the commands that are supposed to be used by agents, coding styles, and so on. These headings are also typically found in the examples provided by the considered tools.[5]

The preliminary analysis presented in this section only scratches the surface, calling for more extensive empirical analyses of the usage "in the wild" of coding agents and their context files in GitHub repositories. Such analyses could start by replicating and extending the results of previous studies [5], [6], [7] at scale. Studying how context files are used and how they evolve over time may provide actionable recommendations for practitioners. Another clear opportunity is to combine our dataset with complementary datasets such as AIDev [4], in order to gain a deeper understanding of how coding agents are reshaping software engineering practices.

VI. Limitations

The dataset construction process was subject to several limitations. We only considered repositories obtained through the SEART search engine [9], limiting to repositories with at least 300 commits, and at least one commit in the last year. While this allowed to exclude experimental, personal and abandoned projects, it may have excluded smaller or less active development projets as well.

As for any analysis focusing on GitHub repositories, we are subject to the well-known perils and pitfalls of git and GitHub [10], [11]. One such peril concerns the possibility of rewriting the git history of a repository. A related limitation is that, given that git repositories may have used git branches and multi-parent commits, it can be hard to establish the exact timeline reflecting the modification histories. We only considered the main branch of each repository, and followed the first-parent rule to traverse commits. While this allows for a linear, flat representation of the histories, it also excludes commits belonging to other branches.

Finally, the configuration we defined for cofee for this dataset is based on a manual analysis of the possible locations of context, skill and subagent files, as described in the documentation of each coding agent. We cannot guarantee completness, since the documentation may have been outdated, incomplete or may simply have changed through time. Conversely, we cannot exclude that some files do not correspond to the categories of context, skill and subagent files and are used for an entirely different purpose. However, we did not encounter any such cases during our manual validation of the dataset.

[5]See https://claude.com/blog/using-claude-md-files for an example.

VII. Conclusion

We presented a large dataset of 142,294 context, skill and subagent file histories used by six major coding agents: Claude, Codex, GitHub Copilot, Gemini, Windsurf and Cursor. The dataset contains 401,873 file revisions, covering 185,795 commits across 27,717 GitHub repositories and spanning a period of nearly two years. We also provide cofee, a generic tool that can be used to further expand this dataset. We start to observe interesting evolutionary patterns in this dataset. We are therefore convinced that the proposed dataset and associated extraction tool will enable the software engineering research community to conduct large-scale longitudinal empirical studies on the evolution of coding agents in GitHub.

VIII. Acknowledgement and Data Availability

This work is supported by F.R.S.-FNRS under grant numbers J.0147.24 and F.4515.23. The dataset presented in this article is available on Zenodo: https://doi.org/10.5281/zenodo.20396488. The cofee tool used to create the dataset is available on https://github.com/sgl-umons/cofee and on PyPI.

References

[1] R. Robbes, T. Matricon, T. Degueule, A. Hora, and S. Zacchiroli, "Promises, perils, and (timely) heuristics for mining coding agent activity," in *Int'l Conf. Mining Software Repositories*. IEEE, 2026.

[2] J. L. Lulla, S. Mohsenimofidi, M. Galster, J. M. Zhang, S. Baltes, and C. Treude, "On the impact of AGENTS.md files on the efficiency of AI coding agents," *preprint arXiv:2601.20404*, 2026.

[3] R. Robbes, T. Matricon, T. Degueule, A. Hora, and S. Zacchiroli, "Agentic much? Adoption of coding agents on GitHub," *ACM Transactions on Software Engineering and Methodology*, 2026.

[4] H. Li, H. Zhang, and A. E. Hassan, "The rise of AI teammates in software engineering (SE) 3.0: How autonomous coding agents are reshaping software engineering," *arXiv:2507.15003*, 2025.

[5] S. Mohsenimofidi, M. Galster, C. Treude, and S. Baltes, "Context engineering for AI agents in open-source software," in *Int'l Conf. Mining Software Repositories*. ACM, 2026.

[6] W. Chatlatanagulchai, H. Li, Y. Kashiwa, B. Reid, K. Thonglek, P. Leelaprute, A. Rungsawang, B. Manaskasemsak, B. Adams, A. E. Hassan, and H. Iida, "Agent READMEs: An empirical study of context files for agentic coding," *preprint arXiv:2511.12884*, 2025.

[7] M. Galster, S. Mohsenimofidi, J. L. Lulla, M. A. Abubakar, C. Treude, and S. Baltes, "Configuring agentic AI coding tools: An exploratory study," in *Int'l Conf. AI-Powered Software*. ACM, 2026.

[8] M. Galster, S. Mohsenimofidi, L. Böhme, J. L. Lulla, M. A. Abubakar, C. Treude, and S. Baltes, "A dataset of agentic AI coding tool configurations," in *Int'l Conf. AI-Powered Software*, 2026.

[9] O. Dabic, E. Aghajani, and G. Bavota, "Sampling projects in GitHub for MSR studies," in *Int'l Conf. Mining Software Repositories*. IEEE, 2021.

[10] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "An in-depth study of the promises and perils of mining GitHub," *Empirical Software Engineering*, vol. 21, no. 5, 2016.

[11] C. Bird, P. C. Rigby, E. T. Barr, D. Hamilton, D. M. German, and P. Devanbu, "The promises and perils of mining git," in *Working Conf. Mining Software Repositories*. IEEE, 2009.