GHAW-H: A Dataset of GitHub Agentic Workflow Histories

Pablo Valenzuela*†, Timo Kehrer*, and Sebastiano Panichella*‡
*University of Bern, Switzerland
†Universidad de La Frontera, Chile
‡Italian Institute of Artificial Intelligence for Industry (AI4I), Italy

***Abstract*—GitHub Agentic Workflows (GH-AW) are an emerging form of repository automation in which developers define agentic workflows in Markdown and execute them through GitHub Actions. However, GH-AW only became publicly available as a Technical Preview in February 2026, and little is known about its early adoption and use in public repositories. To support the systematic study of this emerging practice, we introduce *GHAW-H*, a dataset covering 262 early adopters of GH-AW in public GitHub repositories. The dataset links source Markdown files with their compiled lock files across version histories, providing a foundation for studying how natural-language agentic workflows are defined and evolve in software projects.**

***Index Terms*—GitHub Agentic Workflows, GitHub Actions, agentic automation, natural-language software artifacts, dataset**

I. High-Level Overview

Agentic Workflows are a form of automation that uses Large Language Models (LLMs) to interpret goals, use external tools, and perform actions in their environment [1]. Early LLM-based coding tools focused primarily on assisting developers with code completion and generation. In contrast, recent coding agents can interact directly with repositories and development tools to perform a broader range of software engineering activities [2], [3]. GitHub Agentic Workflows[1] (GH-AW) brings these capabilities to GitHub, enabling developers to define in Markdown how an agent should respond to repository events and execute tasks through GitHub Actions. However, the tool was released only recently, on February 13, 2026, and little is known about its adoption and use.

Figure 1 illustrates how a GH-AW agentic workflow is defined, compiled, and executed within a GitHub repository. In this example, the workflow definition is a Markdown file that combines the configuration in the *frontmatter* with task instructions expressed in natural language. GH-AW compiles this definition into a *lock* file, which runs as a GitHub Actions workflow. During execution, the agent interprets the instructions, analyzes the repository context, and performs the defined task. The workflow may produce observable outcomes through *safe outputs*, such as adding labels, posting comments, or opening a pull request.

To support the systematic study of the early adoption of GH-AW, we introduce *GHAW-H*, a dataset built from 262 repositories that use this tool. *GHAW-H* includes the histories of 604 Markdown files and 2,820 observed versions. Each version preserves the original specification, including the literal *frontmatter*, the natural-language task instructions, and the corresponding compiled snapshot. It also includes the commit SHA, timestamp, position of the version within the file history, and repository-level metadata.

II. Internal Structure

Table I summarizes the structure of *GHAW-H*, including its five normalized tables, record counts, and the GitHub evidence represented in each table. The dataset contains the complete histories of GH-AW files collected from 262 public repositories. These repositories used the tool between its release as a Technical Preview on February 13, 2026, and June 6, 2026, and represent its early adopters [4]. It comprises 604 source Markdown file histories and 2,820 versions. Each version is linked to repository-level metadata. Its representation includes a snapshot of the source Markdown file, a version record containing its commit SHA, timestamp, and position within the file history, and the corresponding compiled lock file snapshot recovered from the same commit.

III. Data Collection and Access

To construct the *GHAW-H* dataset, we first identified repositories that could contain GH-AW files. Candidate repositories were collected using the SEART GitHub Search platform[2], considering repositories with commits since February 13, 2026, the public release date of the GH-AW tool[3]. We then retained repositories containing at least one pair consisting of a Markdown source file (`file.md`) and its compiled workflow (`file.lock.yml`) under `.github/workflows`.

[1]https://github.github.com/gh-aw/

[2]SEART GitHub Search

[3]GitHub announcement

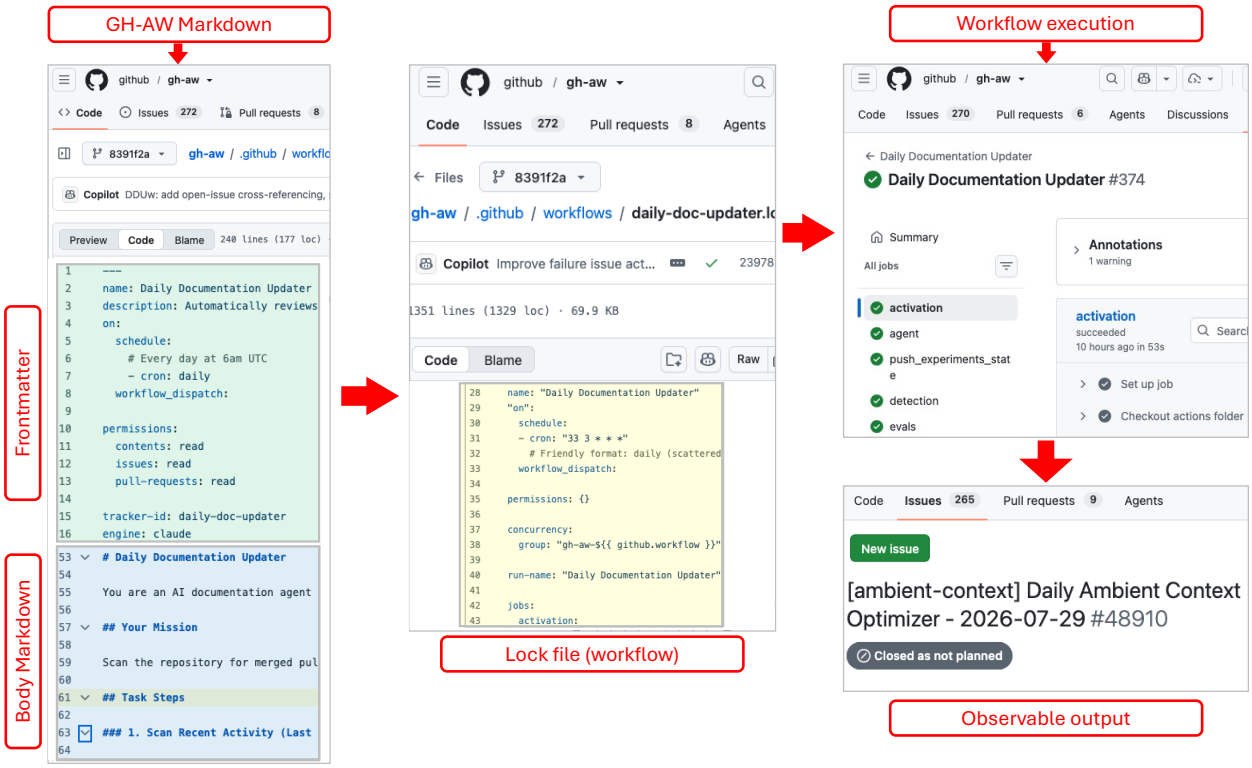

Fig. 1. Overview of a GitHub Agentic Workflow in a repository. A source Markdown file under `.github/workflows/` defines the workflow configuration (Frontmatter) and task instructions (Body Markdown). The GH-AW tooling compiles this source file into a Lock file, which can be executed through GitHub Actions. The execution may produce observable safe outputs, such as a repository comment or a new issue.

TABLE I
OVERVIEW OF THE GH-AW HISTORIES DATASET

	Table	# Records	Content
Repository Metadata	repository	262	Repository-level metadata (name, owner, URL, license, language, stars, forks, and timestamps)
Source History	source_markdown_file_history	604	Histories of confirmed GH-AW source files, including the number of observed versions
	source_markdown_file_snapshot	2,820	Markdown snapshots (path, complete content, literal YAML frontmatter, and natural-language body)
	source_markdown_file_version	2,820	Ordered source-file versions (commit SHA, timestamp, rank, predecessor, and successor)
Compiled Artifacts	lock_file_snapshot	2,820	Commit-aligned compiled `*.lock.yml` snapshots (path, content, and Git blob SHA), one per source version

The *GHAW-H* dataset is publicly available on Hugging Face, where its five Parquet tables can be downloaded and inspected. The release includes four reproducible Jupyter notebooks covering basic dataset loading, general summaries, repository language analysis, and the analysis of source-file evolution histories. These notebooks load the published Parquet files directly from Hugging Face and can be executed through ready-to-use Google Colab links available in the GitHub repository and the Hugging Face dataset card. The release also includes the dataset schema, the table dictionary, and supporting visualizations. The corresponding links are provided below.

- https://pavt.github.io/GHAW-H/
- https://huggingface.co/datasets/pavtch/GHAW-H
- https://zenodo.org/records/22084012
- https://github.com/pavt/GHAW-H

IV. Potential Research Questions

The GHAW-H dataset provides an empirical basis for studying agentic workflow specifications. Its main artifacts are text-based and include natural-language instructions written in Markdown, structured frontmatter declarations, and compiled workflow definitions in YAML. This combination makes Natural Language Processing techniques suitable for analyzing the content of these artifacts and the relationships among them. Building on these possibilities, GHAW-H can support different types of contributions, including empirical studies, tool-oriented papers, comparative analyses with related datasets, and replication or extension studies. In the following, we outline example research questions that can be investigated using such techniques.

A. Instruction Structure and Developer Intent

The natural-language instructions contained in GH-AW Markdown files make it possible to study what developers seek to accomplish through agents and how they express that intent in workflow specifications. In particular, the dataset enables research questions such as: *What software engineering activities do developers delegate to agents through GH-AW?* and *How do the instructions describe the actions to be performed, the software artifacts involved, and the circumstances under which those actions should be carried out?*

B. Instruction Quality and Specification Defects

Because GH-AW relies on natural-language instructions to define agent behavior, the quality of these instructions may directly affect how clearly the intended task is specified. Thus, the dataset can be used to characterize potential specification defects, including ambiguity, incompleteness, and contradictions. In this context, one question that can be investigated is: *What types of specification defects occur in GH-AW instructions, and which linguistic patterns are associated with them?*

C. Instruction–Configuration Consistency

GH-AW specifications combine natural-language instructions with structured frontmatter declarations that define when a workflow is triggered and which permissions and tools are available to the agent. This makes it possible to examine whether the configuration is consistent with the behavior described in the instructions. For example, the dataset enables research questions such as: *To what extent are the permissions, tools, and triggers declared in the frontmatter consistent with the intended behavior described in the instructions?*

D. Specification-to-Workflow Traceability

Each GH-AW source specification is linked to its corresponding compiled *lock* file, allowing the relationship between developer-written instructions and executable workflow elements to be examined. This supports analyses of how instructions are translated into *jobs*, *steps*, conditions, and permissions, as well as the identification of instructions that are only partially reflected in the compiled workflow. A relevant question in this context can be: *To what extent can the elements of a compiled GH-AW workflow be traced back to its source specification?*

E. Workflow Evolution and Maintenance

The version histories captured in GHAW-H make it possible to observe how agentic workflow specifications change over time. These histories can reveal whether modifications affect the intended behavior of the agent, the *frontmatter* configuration, or the compiled workflow, while also distinguishing behavioral changes from edits limited to wording or documentation. This opens questions such as: *How does the intended behavior of GH-AW workflows evolve across successive versions?* and *How do changes in natural-language instructions relate to changes in the frontmatter configuration and compiled workflow?*

V. Related Work

Existing datasets on AI-based coding agents capture the different ways in which these systems operate within software repositories. AIDev collects pull requests created by agents, along with their associated commits, reviews, comments, and issues [5]–[7], whereas AgentPack focuses on code changes co-authored by humans and agents [8]. SWE-chat records real programming sessions, including user prompts, agent responses, tool calls, and code changes [9]. Other datasets collect context files, commands, skills, and subagents, for tools such as Codex, Cursor, and Gemini [10], [11]. Galster et al. [10] collect configuration files used by agentic coding tools. These datasets contain information about agent contributions, interaction trajectories, and the artifacts used to configure general-purpose coding agents. However, none of these datasets covers GH-AW or captures how its workflow specifications evolve over time.

Datasets on GitHub automation have focused primarily on the definitions and executions of conventional workflows. Cardoen et al. [12] collect histories of GitHub Actions workflow files. The dataset GHALogs complements this area with information about runs, steps, and logs [13]. Other studies have constructed run datasets to analyze resource consumption, failures, and reruns in GitHub Actions [14], [15]. These works enable the study of workflow evolution and runtime behavior,

but they focus on conventional workflows or on evidence generated by their runs. In contrast, to the best of our knowledge *GHAW-H* is the first dataset focused on GH-AW.

VI. Impact

GHAW-H can help establish a common foundation for empirical research on GitHub Agentic Workflows at an early stage of their adoption. Its combination of natural-language instructions, structured configuration, compiled artifacts, and version histories enables researchers to examine agentic workflows from complementary perspectives using the same underlying evidence. Studies derived from the dataset may contribute new taxonomies, empirical findings, analysis techniques, and tools for understanding how agentic behavior is specified and maintained in software repositories. These contributions can also inform broader research on natural-language-based software engineering and repository-level software agents.

VII. Acknowledgment

This work was supported by the Italian Institute of Artificial Intelligence for Industry (IDeaLS lab[4], AI4I) within the framework of the IT4LIA AI Factory project, co-funded by the European Union under Grant Agreement No. 101234224.

References

[1] J. Liu, K. Wang, Y. Chen, X. Peng, Z. Chen, L. Zhang, and Y. Lou, "Large language model-based agents for software engineering: A survey," *ACM Transactions on Software Engineering and Methodology*, 2024.

[2] J. Yang, C. Jimenez, A. Wettig, K. Lieret, S. Yao, K. Narasimhan, and O. Press, "Swe-agent: Agent-computer interfaces enable automated software engineering," *Advances in Neural Information Processing Systems*, vol. 37, pp. 50 528–50 652, 2024.

[3] 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 preprint arXiv:2507.15003*, 2025.

[4] GitHub, "GitHub Agentic Workflows are now in technical preview," https://github.blog/changelog/2026-02-13-github-agentic-workflows-are-now-in-technical-preview/, Feb. 2026, gitHub Changelog. Accessed: 2026-06-16.

[5] H. Li, H. Zhang, and A. E. Hassan, "Aidev: studying ai coding agents on github," *arXiv preprint arXiv:2602.09185*, 2026.

[6] R. Kallis, A. Di Sorbo, G. Canfora, and S. Panichella, "Ticket tagger: Machine learning driven issue classification," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2019, pp. 406–409.

[7] P. Valenzuela-Toledo, C. Wu, S. Hernández, A. Boll, R. Machacek, S. Panichella, and T. Kehrer, "Explaining github actions failures with large language models: Challenges, insights, and limitations," in *2025 IEEE/ACM 33rd International Conference on Program Comprehension (ICPC)*, 2025, pp. 286–297.

[8] Y. Zi, Z. Wu, A. Boruch-Gruszecki, J. Bell, and A. Guha, "Agentpack: A dataset of code changes, co-authored by agents and humans," *arXiv preprint arXiv:2509.21891*, 2025.

[9] J. Baumann, V. Padmakumar, X. Li, J. Yang, D. Yang, and S. Koyejo, "Swe-chat: Coding agent interactions from real users in the wild," *arXiv preprint arXiv:2604.20779*, 2026.

[10] 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 *Proceedings of the 3rd ACM International Conference on AI-Powered Software*, 2026, pp. 314–322.

[11] M. Galster, S. Mohsenimofidi, J. L. Lulla, M. A. Abubakar, C. Treude, and S. Baltes, "Configuring agentic ai coding tools: An exploratory study," 2026.

[12] G. Cardoen, T. Mens, and A. Decan, "A dataset of github actions workflow histories," in *Proceedings of the 21st International Conference on Mining Software Repositories*, 2024, pp. 677–681.

[13] F. Moriconi, T. Durieux, J.-R. Falleri, R. Troncy, and A. Francillon, "Ghalogs: Large-scale dataset of github actions runs," in *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*. IEEE, 2025, pp. 669–673.

[14] L. Zheng, S. Li, X. Huang, J. Huang, B. Lin, J. Chen, and J. Xuan, "Why do github actions workflows fail? an empirical study," *ACM Transactions on Software Engineering and Methodology*, vol. 35, no. 5, pp. 1–29, 2026.

[15] J. Huang and B. Lin, "On the reruns of github actions workflows," *ACM Transactions on Software Engineering and Methodology*, 2026.

[4] https://ai4i.it/rde-labs/#IDEALS