gawd: A Differencing Tool for GitHub Actions Workflows

Pooya Rostami Mazrae
pooya.rostamimazrae@umons.ac.be
University of Mons (UMONS)
Mons, Belgium

Alexandre Decan
alexandre.decan@umons.ac.be
F.R.S.-FNRS Research Associate
University of Mons (UMONS)
Mons, Belgium

Tom Mens
tom.mens@umons.ac.be
University of Mons (UMONS)
Mons, Belgium

ABSTRACT

The GitHub social coding platform introduced GitHub Actions as a way to automate different aspects of collaborative software development through the use of workflow files. It is the most popular CI/CD and workflow automation tool for GitHub. To maintain workflow code over time, it is useful to rely on differencing tools to identify the changes made during successive commits. Unfortunately, existing code differencing tools are not able to correctly identify changes made to workflow files. We therefore implemented *gawd*, a syntactic differencing tool for GitHub Actions workflows. The tool is capable of reporting the addition, deletion, modification and move of syntactic components in workflow files, taking into account the specific syntax of workflows. *gawd* has been evaluated on manually classified sets of workflow changes taken from existing commits in 40 different GitHub repositories, and was able to successfully identify these changes. *gawd* is publicly released as an open source Python tool distributed on PyPI.

KEYWORDS

workflow automation, diff tool, software repository mining, GitHub, software changes, software evolution

ACM Reference Format:
Pooya Rostami Mazrae, Alexandre Decan, and Tom Mens. 2024. gawd: A Differencing Tool for GitHub Actions Workflows. In *21st International Conference on Mining Software Repositories (MSR '24), April 15–16, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3643991.3644873

1 INTRODUCTION

GitHub Actions has become the most popular workflow automation tool on GitHub [7, 16, 23]. It facilitates the automation of a diverse set of activities related to software development including testing, quality analysis, continuous integration, dependency management and security monitoring. GitHub Actions can initiate these automated activities in response to specific triggers including commits, comments, issues, pull requests, and scheduled tasks.

Prior research has focused on changes in CI/CD pipelines based on Travis [11, 15, 27, 28]. In a similar vein, researchers have started to explore changes in GitHub Actions workflows [19, 24]. However, such workflows still lack a deep understanding on the types of changes they exhibit, and the frequency of these changes.

The typical way of identifying changes made to code files during commits is through the use of the git diff tool[1] or one of its variants. When applied to GitHub workflow files, however, the use of such a tool is hindered by its lack of precision when identifying changes. More syntax-aware diff tools exist for analysing changes in source code files for specific programming languages, but we are not aware of any diff tool that takes the specific YAML-based syntax of GitHub Actions workflow files into account.

This is why we have developed gawd, a **G**itHub **A**ctions **W**orkflow **D**ifferencing tool. The tool implements a differencing algorithm that enables the comparison of syntactic workflow differences, thereby supporting in-depth analysis of GitHub Actions workflow file changes and how such workflows evolve over time.

After discussing relevant prior research in Section 2, Section 3 provides a motivating example for the relevance of gawd. Section 4 provides installation instructions and implementation details, and explains how to use gawd in practice. Section 5 evaluates the tool and its current limitations. Finally, Section 6 concludes the paper.

2 RELATED WORK

Many differencing tools have been proposed in the past [1, 2, 4, 18, 20, 26], with `diff` probably being the oldest one [18]. Differencing tools can be language-specific, like `Vdiff` for the `Verilog` hardware description language [10], or language-agnostic, like `LHDiff` which is a hybrid differencing approach for tracking source code lines [3]. Specifically in the context of git and GitHub, the `git diff` tool is used [22]. It implements four variants to identify file differences in commits: the default (and fastest) algorithm Myers [21], minimal, patience, and histogram. While these tools are essentially line-based, more advanced differencing tools have been proposed to take into account the tree-like syntactic structure of code files [8, 9, 12–14, 17]. For example, Gumtree [13] aims to compute edit operations that are close to original developer intent. To do so, they represent source code as an abstract syntax tree (AST) to compute the differences by establishing mappings and deducing the edit operations. Despite the variety and abundance of such differencing tools, we are not aware of any such tool having the ability to comprehensively identify differences between GitHub Actions workflow files.

Previous studies [6, 7, 25] have emphasized the importance of understanding how workflow files evolve for uncovering potential maintenance challenges. Rostami Mazrae et al. [19] studied the evolution of workflow files through an analysis of line-based changes, ignoring the specific syntactic or semantic nature of these changes.

MSR '24, April 15–16, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0587-8/24/04…$15.00
https://doi.org/10.1145/3643991.3644873

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

They studied 271,422 unique workflow file snapshots of 65,067 distinct workflow files, observing that around 10% of all changes were mostly cosmetic (such as changes to comments, indentation and pretty printing, and enhancing file readability by adding or changing labels). The remaining 90% of the changes corresponded to actual code changes, and were mostly modifications of existing code lines, or additions of new lines. Line removals were considerably less frequent with only 3.45% of the workflow files having line removals as their only change and 22.78% of the workflow files changes including line removals together with other types of changes.

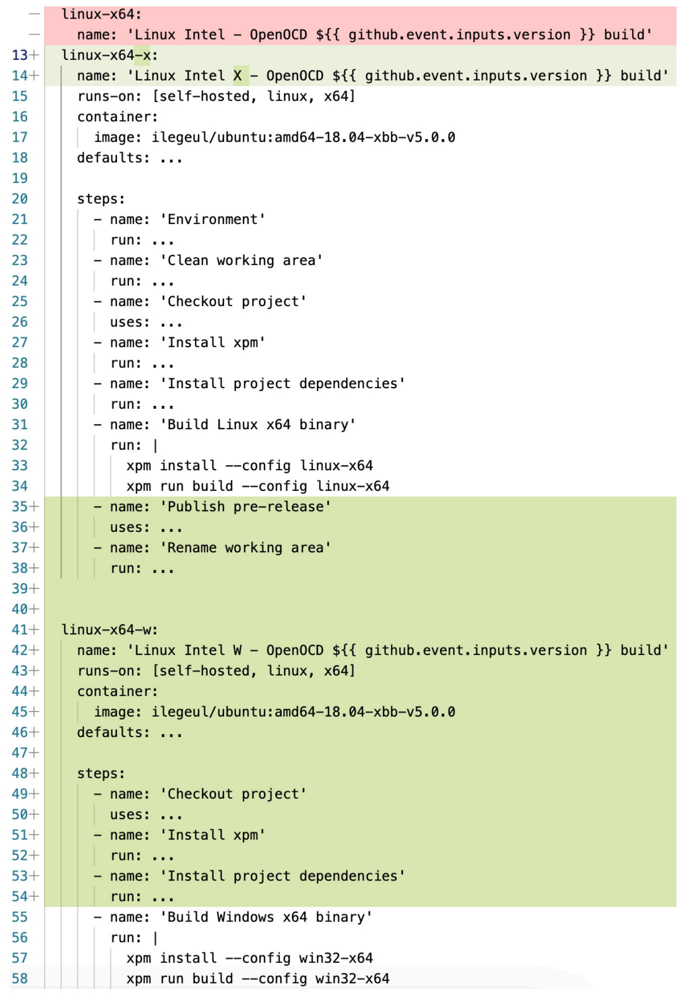
The primary objective of gawd is to enable a more fine-grained syntax-based comprehension of the changes that are being made to GitHub Actions workflow files.

3 MOTIVATING EXAMPLE

Just like any other software artifact, workflow files are susceptible to change during their lifetime. Neither GitHub's default diff tool nor any of git diff algorithms is capable of correctly identifying the set of changes made to these files in the different commits touching them. This is illustrated in the example of Figure 1, depicting the visual diff of the changes made to a workflow file in some commit. [2]

Lines 13-14 (and the two preceding unnumbered lines in red) show a change of job id from linux-x64 to linux-x64-x and a modification of the job's name. Lines 35-36 and 37-38 suggest that a sixth and seventh step[3] have been added to this job, but this is not what happened in reality. Instead, the step named 'Build Windows x64 binary' on lines 55-58 (which is reported by the diff tool as not being part of any change) has been removed from this job, and added into a new job with id linux-x64-w (lines 41-58). As a consequence, the two steps on lines 35-38 have changed their relative position from 7 and 8, respectively, to 6 and 7, respectively. It is clear that the diff tool misses out on a lot of syntactic changes by considering lines 35-54 as a single composite change of 20 consecutive lines being added, while in fact it should be considered as multiple, more primitive changes.

Listing 1 provides a condensed summary of what an improved syntax-aware diff tool should produce for this specific workflow file changeset. Line 1 corresponds to the change of job id and line 2 represents the change of the job's name. Line 3 reflects the fact that the step 'Build Windows x64 binary' that used to be at position 6 is now removed from the job (because it has been inserted as part of a newly added job on line 6, of which the full contents is not shown for the sake of brevity). Lines 4 and 5 correspond to the resulting reordering of the next two steps, that change their relative position in the job because of the removal of the step on line 3.



```
   —  linux-x64:
   —    name: 'Linux Intel - OpenOCD ${{ github.event.inputs.version }} build'
13+   linux-x64-x:
14+     name: 'Linux Intel X - OpenOCD ${{ github.event.inputs.version }} build'
15      runs-on: [self-hosted, linux, x64]
16      container:
17        image: ilegeul/ubuntu:amd64-18.04-xbb-v5.0.0
18      defaults: ...
19
20      steps:
21        - name: 'Environment'
22          run: ...
23        - name: 'Clean working area'
24          run: ...
25        - name: 'Checkout project'
26          uses: ...
27        - name: 'Install xpm'
28          run: ...
29        - name: 'Install project dependencies'
30          run: ...
31        - name: 'Build Linux x64 binary'
32          run: |
33            xpm install --config linux-x64
34            xpm run build --config linux-x64
35+       - name: 'Publish pre-release'
36+         uses: ...
37+       - name: 'Rename working area'
38+         run: ...
39+
40+
41+   linux-x64-w:
42+     name: 'Linux Intel W - OpenOCD ${{ github.event.inputs.version }} build'
43+     runs-on: [self-hosted, linux, x64]
44+     container:
45+       image: ilegeul/ubuntu:amd64-18.04-xbb-v5.0.0
46+     defaults: ...
47+
48+     steps:
49+       - name: 'Checkout project'
50+         uses: ...
51+       - name: 'Install xpm'
52+         run: ...
53+       - name: 'Install project dependencies'
54+         run: ...
55        - name: 'Build Windows x64 binary'
56          run: |
57            xpm install --config win32-x64
58            xpm run build --config win32-x64
```

Figure 1: Visual diff of the changes made to some GHA workflow during a commit to a GitHub repository.

Listing 1: Expected output.

```
1 renamed jobs.linux-x64 to jobs.linux-x64-x
2 changed jobs.linux-x64.name from "Linux␣Intel␣..." to"..."
3 removed jobs.linux-x64.steps[6]
4 moved jobs.linux-x64.steps[7] to jobs.linux-x64-x.steps[6]
5 moved jobs.linux-x64.steps[8] to jobs.linux-x64-x.steps[7]
6 added jobs.linux-x64-w with "..."
```

4 GAWD

4.1 Installation and execution

gawd has been developed as a standalone Python 3 package with a command-line interface. gawd reports on the syntactic differences between two GitHub Actions workflow files. Given two workflow files, gawd identifies the addition or removal of steps and key-value pairs, changes in values associated to a given key, and moves of steps to a different position within an existing job. gawd is publicly available on PyPI and can be installed through pip (pip install gawd), and its source code is available on https://github.com/pooya-rostami/gawd.

To compute the syntactic diff between two workflow YAML files `file1.yml` and `file2.yml`, one simply needs to run the command `gawd file1.yml file2.yml`

The tool can be configured with a range of optional arguments. For all details, we refer the reader to the tool's built-in help (i.e.,

[2]Original commit: https://github.com/d99kris/nchat/commit/0438344525f6bb36f41c0a145dfe54fad5a3f2c2
[3]We assume a zero-based indexing for steps.

`gawd --help`). For example, the following command outputs the result in JSON format (`--json`) using a more condensed form (`--short`) by not showing the full details of the values associated to each change, as values can sometimes be very long:

```
gawd file1.yml file2.yml --json --short
```

Other optional command-line arguments can be used to change the default behaviour of the tool: *threshold*, *position_weight* and *job_name_weight*. *threshold* can be set between 0 and 1 (by default it is 0.5) to modify the sensitivity of the tool in identifying changes. A higher threshold results in more instances of identified changes, renames or moves, while a lower threshold favors additions and removals. *position_weight* specifies a value between 0 and 1 (by default it is 0.2) that determines the weight assigned to the relative positions of two items. Increasing this value increases the importance of the relative position of two items in a sequence (e.g., two steps in a job) while decreasing the influence of their content (dis)similarity. *job_name_weight* specifies a value between 0 and 1 (by default it is 0.2) that quantifies the significance of differences in job names when determining the similarity of jobs. A higher value places greater importance on job name similarity and reduces the influence of their content (dis)similarity.

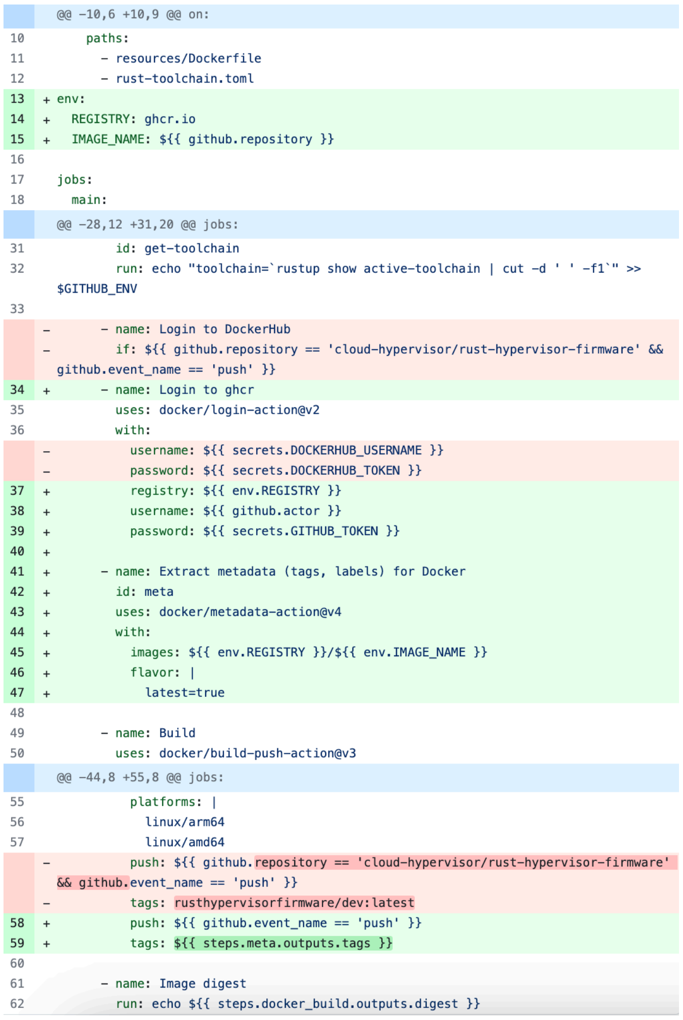
4.2 Implementation details

The implementation of gawd relies on a set of functions dedicated to finding matches, calculating distances and comparing differences.

Finding matches. The two functions *find_list_matches* and *find_job_matches* are dedicated to identifying and computing matches, whether they involve items within sequences or jobs. These functions assign distance scores to various item combinations, allowing for the generation of sorted lists of matches, while considering positions and job names. In this process, all potential matches are returned, ordered by distance, but each item is only matched once. Items whose distance is below the given threshold are categorized as *changed*. Other items are categorized as *removed* or *added* depending on their presence in the first (old) or second (new) file.

Calculating distances. The *dict_distance* function computes a normalized distance between dictionaries, based on common, changed, added, and removed items. The *distance* function calculates distances between objects, accounting for data types such as simple values, dictionaries (by relying on *dict_distance*) and sequences. This comprehensive approach enables a nuanced understanding of similarity and difference of two data structures.

Comparing differences. Function *diff_workflows* takes two workflow files represented as dictionaries, performs a comparison, and generates a list of differences. These differences are categorized into additions, removals, renames, changes, and moves with associated paths and values from both the old and new workflows. Function *diff_workflow_files* operates on two workflow files specified by their file paths. It reads and converts these files into dictionaries and calls *diff_workflows* to provide a list of differences between the two workflow files.



```
@@ -10,6 +10,9 @@ on:
      paths:
        - resources/Dockerfile
        - rust-toolchain.toml
+ env:
+   REGISTRY: ghcr.io
+   IMAGE_NAME: ${{ github.repository }}

  jobs:
    main:
@@ -28,12 +31,20 @@ jobs:
          id: get-toolchain
          run: echo "toolchain=`rustup show active-toolchain | cut -d ' ' -f1`" >>
  $GITHUB_ENV

-       - name: Login to DockerHub
-         if: ${{ github.repository == 'cloud-hypervisor/rust-hypervisor-firmware' &&
  github.event_name == 'push' }}
+       - name: Login to ghcr
          uses: docker/login-action@v2
          with:
-           username: ${{ secrets.DOCKERHUB_USERNAME }}
-           password: ${{ secrets.DOCKERHUB_TOKEN }}
+           registry: ${{ env.REGISTRY }}
+           username: ${{ github.actor }}
+           password: ${{ secrets.GITHUB_TOKEN }}
+
+       - name: Extract metadata (tags, labels) for Docker
+         id: meta
+         uses: docker/metadata-action@v4
+         with:
+           images: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}
+           flavor: |
+             latest=true

        - name: Build
          uses: docker/build-push-action@v3
@@ -44,8 +55,8 @@ jobs:
            platforms: |
              linux/arm64
              linux/amd64
-           push: ${{ github.repository == 'cloud-hypervisor/rust-hypervisor-firmware'
  && github.event_name == 'push' }}
-           tags: rusthypervisorfirmware/dev:latest
+           push: ${{ github.event_name == 'push' }}
+           tags: ${{ steps.meta.outputs.tags }}

        - name: Image digest
          run: echo ${{ steps.docker_build.outputs.digest }}
```

Figure 2: Real-world example of GitHub's visual diff of workflow changes made in a commit.

5 EVALUATION

5.1 Example

Let us consider a concrete example of a diff between two versions of a workflow file associated with a specific commit.[4] Figure 2 shows the diff as reported by GitHub. In contrast, Figure 3 illustrates gawd's output, presenting the changes in a more fine-grained and syntax-aware manner. It highlights gawd's proficiency in recognizing step movements (in terms of their relative position within a job) resulting from the addition or deletion (in this case addition) of steps preceding them in the updated workflow file. Additionally, the tool successfully captures changes in the values corresponding to each key within the workflow files. For each such change, the fully qualified path related to the key is reported. This capacity to

[4]We used the following commit for this example: https://github.com/cloud-hypervisor/rust-hypervisor-firmware/commit/5e2934f2d9cb52c54e696bccc687d52f0ee402f5#diff-a9b4f1a51dd81dfaef1e9d563dfe6045ab8089a05200453d6e0b2d66201f419f

```
added env with {'REGISTRY': 'ghcr.io', 'IMAGE_NAME': '${{ github.repository }}'}
moved jobs.main.steps[6] to jobs.main.steps[7]
moved jobs.main.steps[5] to jobs.main.steps[6]
changed jobs.main.steps[5].with.push from "${{ github.repository == ... }}" to "${{ github.event_name == 'push' }}"
changed jobs.main.steps[5].with.tags from 'rusthypervisorfirmware/dev:latest' to '${{ steps.meta.outputs.tags g}}'
removed jobs.main.steps[4].if with "${{ github.repository == 'cloud-hypervisor/rust-hypervisor-firmware' ... }}"
changed jobs.main.steps[4].name from 'Login to DockerHub' to 'Login to ghcr'
added jobs.main.steps[4].with.registry with '${{ env.REGISTRY }}'
changed jobs.main.steps[4].with.username from '${{ secrets.DOCKERHUB_USERNAME }}' to '${{ github.actor }}'
changed jobs.main.steps[4].with.password from '${{ secrets.DOCKERHUB_TOKEN }}' to '${{ secrets.GITHUB_TOKEN }}'
added jobs.main.steps[5] with {'name': 'Extract metadata (tags, labels) for Docker', 'id': 'meta', ... }
```

Figure 3: gawd output for the example of Figure 2.

trace changes to the exact location in the workflow's hierarchical structure enhances the tool's utility for software developers and engineers in effectively interpreting workflow file modifications.

5.2 Validation

We validated the correctness of gawd by manually checking the tool's output for real-life cases of workflow file changes made in existing commits in 40 different GitHub repositories. To do so, we first used the SEART GitHub search engine [5] to create a dataset comprising GitHub repositories that (i) are not forks of existing repositories; (ii) use GitHub Actions and contain at least one workflow file in the `.github/workflow` directory; (ii) have been created after 2019-01-01 and before 2019-06-01; (iii) have at least 100 stars and 300 commits; and (iv) have their most recent commit after 2023-01-01. We randomly selected 40 of these repositories and locally cloned them on 2023-05-24.

For each of these 40 repositories, we extracted all commits involving changes to GitHub Actions workflow files and randomly selected one such commit. Using the 40 selected commits we manually checked whether gawd's output corresponded to our own interpretation of the changes being made to the workflow. In those cases where the commit contained changes to multiple workflow files, we selected the first workflow file.

The manual confirmation of the correctness of the tool's output proceeded as follows. Next, all three researchers discussed together about the interpretation of their changes, in order to come to a consensus on the actual changes that were being observed in the workflow files. These changes were then compared against gawd's output. The outcome of this comparison revealed that gawd successfully identifies all code-related modifications within workflow files present in our test cases.

We have integrated this manual confirmation of gawd's output on 40 cases of workflow file changes as an automated test suite that is part of gawd's continuous integration process whenever we are changing its implementation and creating new versions and releases.

5.3 Limitations

A limitation of gawd is its inability to properly comprehend composite "semantic" changes such as merging (or conversely splitting) multiple steps into a single one, moving steps between jobs, replacing a step executing some shell commands (`run:` key) by a step using a reusable Action (`uses:` key) and vice versa. Similarly, gawd ignores changes to comments, or cosmetic changes such as the addition or removal of empty lines, whitespaces or indentations that do not affect the workflow syntax.

In future endeavors, we plan to enhance gawd's functionality by implementing a three-way diff feature, facilitating improved merging of workflow files within repositories. We also aim to provide a visualization to allow users to identify the reported changesets in a familiar way, as in traditional line-based diff tools. Furthermore, similar to the Unix patch tool, we plan to create a tool taking as input some changes generated by gawd and to apply them to similar workflow files. We also plan to make gawd aware of the various programming languages that can be used in workflow files (e.g., shell commands or Python scripts in `run:`) when computing the distance between values.

6 CONCLUSION

This paper presented gawd, a differencing tool designed to report syntactic changes between pairs of GitHub Actions workflow files. The tool was implemented as an open source Python library with a command-line interface. We manually checked the accuracy of gawd on 40 different commits involving workflow file modifications, representing real-life scenarios involving the addition, removal, modification and move of workflow code fragments.

We plan to utilize gawd as a basis for carrying out large-scale empirical quantitative analyses of GitHub Actions workflow evolution, focusing on changes in the configuration, execution, quality-related and security-related aspects of workflow code. Through such analysis we aim to get insight in, and provide support for, the challenges encountered by workflow maintainers. For example, identifying frequent changes could lead to a system for recommending changes or automating refactorings.

ACKNOWLEDGMENTS

This work is supported by research project ARC-21/25 UMONS3 Action de Recherche Concertée financée par le Ministère de la Communauté française - Direction générale de l'Enseignement non obligatoire et de la Recherche scientifique, and F.R.S.-FNRS under grant numbers J.0147.24, T.0149.22 and F.4515.23.

REFERENCES

[1] Raihan Al-Ekram, Archana Adma, and Olga Baysal. 2005. diffX: an algorithm to detect changes in multi-version XML documents. In *Conference of the Centre for Advanced Studies on Collaborative research*. Citeseer, 1–11.

[2] Taweesup Apiwattanapong, Alessandro Orso, and Mary Jean Harrold. 2004. A differencing algorithm for object-oriented programs. In *International Conference on Automated Software Engineering (ASE)*. IEEE, 2–13.

[3] Muhammad Asaduzzaman, Chanchal K Roy, Kevin A Schneider, and Massimiliano Di Penta. 2013. Lhdiff: A language-independent hybrid approach for tracking source code lines. In *International Conference on Software Maintenance (ICSM)*. IEEE, 230–239.

[4] Sudarshan S Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom. 1996. Change detection in hierarchically structured information. *ACM SIGMOD Record* 25, 2 (1996), 493–504.

[5] Ozren Dabic, Emad Aghajani, and Gabriele Bavota. 2021. Sampling projects in GitHub for MSR studies. In *International Conference on Mining Software Repositories*.

[6] Alexandre Decan, Tom Mens, and Hassan Onsori Delicheh. 2023. On the outdatedness of workflows in the GitHub Actions ecosystem. *Journal of Systems and Software* 206 (2023), 111827.

[7] Alexandre Decan, Tom Mens, Pooya Rostami Mazrae, and Mehdi Golzadeh. 2022. On the Use of GitHub Actions in Software Development Repositories. In *Int'l Conf. Software Maintenance and Evolution*.

[8] Sep Dehpour. [n. d.]. *DeepDiff*. https://github.com/seperman/deepdiff

[9] Georg Dotzler and Michael Philippsen. 2016. Move-optimized source code tree differencing. In *International Conference on Automated Software Engineering (ASE)*. 660–671.

[10] Adam Duley, Chris Spandikow, and Miryung Kim. 2012. Vdiff: a program differencing algorithm for Verilog hardware description language. *Automated Software Engineering* 19, 4 (2012), 459–490.

[11] Thomas Durieux, Rui Abreu, Martin Monperrus, Tegawendé F Bissyandé, and Luís Cruz. 2019. An analysis of 35+ million jobs of Travis CI. In *Int'l Conf. Software Maintenance and Evolution (ICSME)*.

[12] Fatih Erikli. [n. d.]. *dictdiffer*. https://github.com/inveniosoftware/dictdiffer

[13] Jean-Rémy Falleri, Floréal Morandat, Xavier Blanc, Matias Martinez, and Martin Monperrus. 2014. Fine-grained and accurate source code differencing. In *International Conference on Automated Software Engineering (ASE)*. 313–324. https://doi.org/10.1145/2642937.2642982

[14] Veit Frick. 2020. Understanding software changes: Extracting, classifying, and presenting fine-grained source code changes. In *International Conference on Software Engineering: Companion Proceedings*. 226–229.

[15] Keheliya Gallaba and Shane McIntosh. 2018. Use and misuse of continuous integration features: An empirical study of projects that (mis) use Travis CI. *Trans. Software Engineering* 46, 1 (2018).

[16] Mehdi Golzadeh, Alexandre Decan, and Tom Mens. 2022. On the rise and fall of CI services in GitHub. In *Int'l Conf. Software Analysis, Evolution and Reengineering (SANER)*.

[17] Kaifeng Huang, Bihuan Chen, Xin Peng, Daihong Zhou, Ying Wang, Yang Liu, and Wenyun Zhao. 2018. Cldiff: generating concise linked code differences. In *International Conference on Automated Software Engineering (ASE)*. 679–690.

[18] James W. Hunt and M. Douglas McIlroy. 1976. *An Algorithm for Differential File Comparison*. Technical Report 41. Computing Science Technical Report, Bell Laboratories.

[19] Pooya Rostami Mazrae, Alexandre Decan, Tom Mens, and Mairieli Wessel. 2023. A Preliminary Study of GitHub Actions Workflow Changes. In *Seminar on Advanced Techniques and Tools for Software Evolution (SATToSE)*, Vol. 3483. CEUR Workshop Proceedings.

[20] Webb Miller and Eugene W. Myers. 1985. A file comparison program. *Software: Practice and Experience* 15, 11 (1985), 1025–1040.

[21] Eugene W. Myers. 1986. An O(ND) difference algorithm and its variations. *Algorithmica* 1, 1-4 (1986), 251–266.

[22] Yusuf Sulistyo Nugroho, Hideaki Hata, and Kenichi Matsumoto. 2020. How different are different diff algorithms in Git? Use–histogram for code changes. *Empirical Software Engineering* 25 (2020), 790–823.

[23] Pooya Rostami Mazrae, Tom Mens, Mehdi Golzadeh, and Alexandre Decan. 2023. On the usage, co-usage and migration of CI/CD tools: A qualitative analysis. *Empirical Software Engineering* 28, 2 (2023), 52.

[24] Pablo Valenzuela-Toledo and Alexandre Bergel. 2022. Evolution of GitHub Action Workflows. In *Int'l Conf. Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 123–127.

[25] Mairieli Wessel, Tom Mens, Alexandre Decan, and Pooya Rostami Mazrae. 2023. *The GitHub Development Workflow Automation Ecosystems*. Springer International Publishing, Cham, 183–214. https://doi.org/10.1007/978-3-031-36060-2_8

[26] Zhenchang Xing and Eleni Stroulia. 2005. UMLDiff: an algorithm for object-oriented design differencing. In *International Conference on Automated Software Engineering (ASE)*. 54–65.

[27] Fiorella Zampetti, Salvatore Geremia, Gabriele Bavota, and Massimiliano Di Penta. 2021. CI/CD pipelines evolution and restructuring: A qualitative and quantitative study. In *Int'l Conf. Software Maintenance and Evolution (ICSME)*. IEEE.

[28] Fiorella Zampetti, Carmine Vassallo, Sebastiano Panichella, Gerardo Canfora, Harald Gall, and Massimiliano Di Penta. 2020. An empirical characterization of bad practices in continuous integration. *Emp. Soft. Eng.* 25 (2020).