Evolution of GitHub Action Workflows

Pablo Valenzuela-Toledo
ISCLab, DCC, University of Chile
DCI, Universidad de La Frontera
pvalenzue@dcc.uchile.cl

Alexandre Bergel
ISCLab, DCC
University of Chile
abergel@dcc.uchile.cl

***Abstract*—GitHub Actions are an event-driven tool to automatically respond to particular GitHub events. Typical events are receiving new pull requests or publishing a software release. Despite the massive and quick adoption of GitHub Actions, little is known about the incremental construction of GitHub Actions workflow by practitioners.**

This paper presents the result of a manual inspection of 222 commits of GitHub Actions workflows obtained from 10 popular open-source repositories. Our hierarchical taxonomy, obtained by systematically categorizing and tagging workflow modifications, reveals 11 types of modifications and presents opportunities for improvement in the way workflows are built and edited. In particular, our results highlight the need for adequate tooling to support refactoring, debugging and code editing of GitHub Actions workflows.

***Index Terms*—GitHub Actions, Workflow Modifications Taxonomy, Workflow Evolution**

I. Introduction

Automating repetitive tasks of a software development process is nowadays frequently supported by Social Code Platforms, such as GitHub [1]–[3]. GitHub Actions (GA) are a service offered by GitHub to automate all software workflow, including building, testing, and deploying directly from GitHub[1]. GA is relatively new since a beta release of GA was available in November 2019. Since then, GA is now a central service for both practitioners and cloud adoption[2]. Software developers have a positive perception of GA [3].

Despite the relevance of GA in state-of-the-art software development practices, little is known about how practitioners build and maintain GitHub Actions workflows. In particular, it is not clear how practitioners cope with the particularities of developing GA workflows (e.g., workflow are executed only by pushing a change in the repository, debugging a GA workflow is carried out by inspecting logs, workflows are typically edited through a generic text editor through the GitHub interface).

This paper provides a first approximation of how practitioners build and maintain GA workflows. We have revised the content and history of 10 open-source GitHub repositories. The repositories were selected from a previous related effort [4]. From the selected projects, we identified 222 commits that directly involve the edition of at least one GA workflow. Our taxonomy comprises 11 different types of workflow modifications, themselves classified into the following categories: (i) file modification, (ii) execution configuration, and (iii) workflow construction.

Our results highlight a number of deficiencies in the way GA workflows are produced and maintained. In particular, we found 4.5% of modifications are about fixing the YAML syntax used in workflow, another 5.86% involve debugging GA workflows.

The outline of the paper is as follows: Section II succinctly describes GitHub Actions as theoretical background; Section III states a research question and presents the methodology we used to build a taxonomy; Section IV presents our results; Section V presents the threats to our effort; Section VII concludes and highlights some of our future works.

II. Workflows Modifications in GitHub Actions

GitHub Actions (GA) is an event-driven feature provided by the GitHub Social coding platform to automate software development tasks. Using workflows as an essential building block, the configuration of actions automatically trigger responses to different events, which is the base for continuous integration. These events reflect specific software development activities that trigger the workflow execution. For example, a workflow running all the unit tests may be executed upon merging a pull request into a particular branch. Practitioners can define actions by creating entirely new workflows or reuse previously built actions publicly available in the GitHub Actions Marketplace[3].

Workflows are stored as scripts in the `.github/workflows` directory and use YAML syntax, and file extension `.yml` or `.yaml`. The lifecycle of a GA workflow is very different from traditional software program. In particular, the execution of a workflow is triggered with any commit in the GitHub repository, commit being related or not to the workflow. As a consequence, the history of GitHub repositories having actions contains great details about all the changes made on workflows, at a fine grain. As any kind of software artifacts, GA workflow files are modified over time, expressing new continuous integration requirements. In our case, we consider a workflow modification as any change in a YAML file.

III. Study Design

This paper report our finding on GA workflow modifications from a qualitative perspective. We propose a taxonomy of workflow modifications that developers tend to perform when using GA. Our study addresses the following research question (RQ):

[1]https://github.com/features/actions

[2]https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/scenarios/github-velocity/

[3]https://github.com/marketplace?type=actions

- **RQ**: What types of GitHub Actions workflows modifications are performed by developers on our set of selected software repositories?

This RQ aims at analyzing a sample and determining the types of workflow modifications carried out by software developers when building/maintaining workflows. Knowing the types of workflow modifications performed by developers is a valuable source of data that can guide the future development of tools to ease the evolution of workflows.

A. Data Collection

To answer our RQ, we mine the commits from ten open-source software GitHub repositories from a previous related effort [4]. We collected these commits in July 2021 and represented the context of our study. The data selection criteria were: (1) commits from repositories using GitHub Actions; (2) commits associated with GitHub Action workflows modifications; and (3) commits that were available between the release of GitHub Actions, and July 30, 2021.

We built our dataset in three sequential steps: (i) we clone the software repositories; (ii) we extract commits that fulfill our data selection criteria; (iii) for each commit we defined a link to the GitHub single commit view; and (iv) we extract the result of the workflow execution. In this way, we take advantage of GitHub to visualize the GA workflow modification.

A total of 222 commits satisfied our selection criteria, representing 5.6% of the total available commits in the period of our study. All 222 workflow modifications are available in our dataset file [5].

B. Methodology

We use the open card sorting approach in our study [6]. Card sorting is a widely used technique in software engineering useful to derive taxonomies from data [7]. In our case, card sorting helps us to categorize GA workflow modifications. Figure 1 illustrates our methodology.

We executed card sorting in three phases, according to guidelines defined by Few [6]. In the preparation phase, we configured our cards using descriptive coding [8], which is a useful technique to generate a rich description of the study subject. We wrote down an explicit fine-grain description of each workflow modification using the GitHub single commit view. We adopt the criteria of including an explanation about why and where the change occurred. We added a total of 222 descriptions (one for each workflow-related commit). Then, we physically printed out our 222 cards, each including a description and context information of each GA workflow modification (project name, project owner, file name, identification number).

In the execution phase, we label and sort each card into meaningful groups with a descriptive title (e.g., the card with text *"A new workflow was added, for push event"* was classified as New Workflow). Finally, in the analysis phase, with no predefined groups we derive abstract hierarchies in order to deduce general categories.

We adopt our card sorting approach considering the following execution criteria:

- The description criteria of the preparation phase was piloted by 3 participants, 2 undergrad students and the first author. Here, we assigned 50 workflow modifications links to each participant to write down respective descriptions. Participants reviewed and compared the descriptions and derived the final description structure.
- The card labelling during the execution phase was supported by the physical creation of printed cards. Each card includes related data, for a better comprehension of the workflow modification.
- We manually build a hierarchy and sort all the categories accordingly.

IV. Results Discussion

We built a taxonomy consisting of 11 types of GA workflow modifications including a total of 222 labelled modifications (see Table I). We grouped these modifications into 3 main categories: (i) file modification; (ii) execution configuration; and (iii) workflow construction.

We present representative examples for each category and discuss implications for practitioners derived from our findings. **The whole list of commits is available in our accompanying artifact [5]**.

TABLE I: Workflow Modifications Taxonomy.

Primary group	Secondary group	#	%
File Modification	New workflow	23	10.36
	Deleted workflow	1	0.45
	Workflow file-name modification	4	1.80
Execution Configuration	Showing instructions execution	2	0.90
	Updated instruction	10	4.50
	Debugging	13	5.85
Workflow Construction	Instruction definition	134	60.36
	Code review configuration	7	3.15
	Commented code	2	0.90
	Code indentation	10	4.50
	Testing configuration	16	7.20
Total modifications		222	100%

A. File Modifications

A total of 27 (12.16%) workflow modifications are related to file modifications. This category covers the following types of modifications: (i) New workflow; (ii) Deleted workflow; and (iii) Workflow file-name modification.

New workflow (23). New workflow category refers to adding a new workflow to the repository. Practitioners incorporate new workflows as a way to automate answers to events. We found 23 samples of this type of modifications (e.g., [9]).

Deleted workflow (1). Deleted workflow refers to removing an existing workflow from the repository. Such a removal typically happens to reduce the activity on GitHub's cloud infrastructure, to remove obsolete workflows, or to keep the activity under a GitHub free plan (2000 minutes per month). We found 1 sample of this category (e.g., [10]).

Workflow file-name modification (4). This category covers any change of the workflow file name to reflect refactoring. We found 4 samples in this category (e.g., [11]).

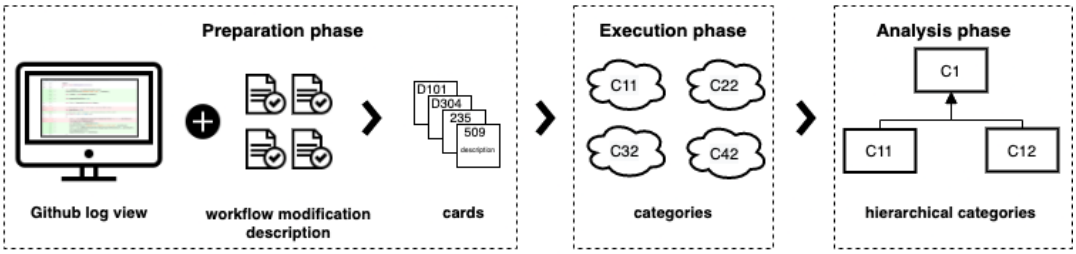

Fig. 1: Illustration of our methodology.

Discussion. Our results highlight modifications related to modifications of the workflows files. In this category the modification type most frequently found is, not surprisingly, New Workflow. Indeed, as it happens with new added source code, new workflows are frequently added to implement new features, e.g., a greeting answer to a push event [12]. We also notice that new workflows are part of migration from previous similar used tools. Here, using the related pull request discussion [13] of the modification [14], we verify a migration from Travis to GitHub Actions. However, adding workflows might not be trivial since not all new ones are correctly executed. We found that just 29% of newly added workflows were executed successfully. We also found a similar scenario in Deleted Workflow category, as we can notice in the related pull request discussion [15] of the modification [10].

These results are consistent with previous studies regarding the incorporation of new automation practices in the software development process. The mere introduction of an automated build infrastructure in the development workflow does not guarantee to achieve the expected benefits of automation such as faster releases [16] and increased developer productivity [17].

B. Execution Configuration

A total of 25 (11.26%) workflow modifications are related to execution configuration, including: (i) Showing instructions execution; (ii) Updated instruction; and (iii) Debugging.

Showing instructions execution (2). Showing instructions execution category covers modifications that allows visualizing the execution of one or more workflow instructions. We found 2 samples in this category (e.g., Figure 2 [18]).

2 .github/workflows/continuous.yml

```
@@ -32,6 +32,8 @@ jobs:
32 32
33 33        - name: set-version
34 34          run: |
   35 +          echo $SMALLTALK_CI_VM
   36 +          ls $SMALLTALK_CI_VM
35 37          ./$SMALLTALK_CI_VM/pharo $PROJECT_NAME.image --save eval "Transcript crShow: $GITHUB_SHA"
36 38
37 39        - name: rename
```

Fig. 2: Adding instructions to show workflow execution [18].

Updated Instruction (10). This category refers to modifications where one or more changes are done to update tools/software used in the workflow. We found 10 samples of this category (e.g., Figure 3 [19]).

2 .github/workflows/black.yml

```
@@ -10,7 +10,7 @@ jobs:
10 10        steps:
11 11        - uses: actions/checkout@v2
12 12        - name: Set up Python 3.8
13    -        uses: actions/setup-python@v1
   13 +        uses: actions/setup-python@v2
14 14          with:
15 15            python-version: 3.8
16 16        - name: Install Black
```

Fig. 3: Updating an action from version 1 to version 2 [19].

Debugging (13). Debugging category refers to modifications made to find or reduce the number of defects of the workflow. Practitioners often debug their workflows to make them behave as expected [20]. We found 13 samples of this category (e.g., Figure 4 [21]).

13 .github/workflows/flake8.yml

```
@@ -17,18 +17,7 @@ jobs:
17 17          uses: actions/setup-python@v2
18 18          with:
19 19            python-version: ${{ matrix.python-version }}
20    -      - name: Install dependencies
21    -        run: |
22    -          python -m pip install --upgrade pip
23    -          pip install -r requirements.txt
24 20        - name: Lint with flake8
25 21          run: |
26 22            pip install flake8
27    -          # stop the build if there are Python syntax errors or undefined names
28    -          flake8 . --count --select=E9,F63,F7,F82 --show-source --statistics
29    -          # exit-zero treats all errors as warnings.
30    -          flake8 . --exit-zero
31    -      - name: Test with pytest
32    -        run: |
33    -          pip install pytest
34    -          pytest
   23 +          flake8 .
```

Fig. 4: Deliberately making the workflow fail [21].

Discussion. Most of the modifications related to execution configuration can be generalized to issues experienced by developers when executing workflows. This scenario is strongly related to the lack of cultural shift that involves incorporate automation tools [22]. For example, since GA debugging build failures happen on a remote server with limited access, fixing a

bug is not a straightforward task. Also, waiting for the results of long builds threatens developer productivity. Here, we notice that the workflow creation-edition-execution process is not tool-supported and explains why nearly the 40% of the total modifications we identified do not execute correctly.

C. Workflow Construction

A total of 169 (76.13%) workflow modifications are related to workflow construction. This category considers modifications related to: (i) Instruction definition; (ii) Code review configuration; (iii) Commented code; (iv) Code indentation; (v) testing configuration.

Instruction definition (134). This category refers to adding, modifying an instruction at any level (e.g. at job or step level) in the workflow. We found 134 samples of this category (e.g., Figure 5 [23]).

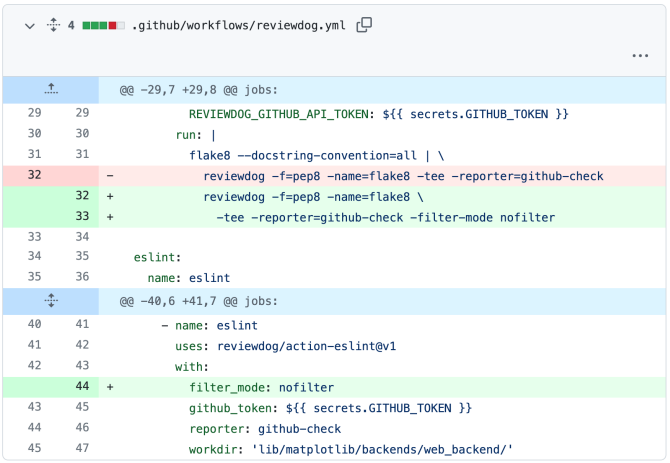

Fig. 5: Defining workflow instructions to add new tool specifications [23].

Code review configuration (7). This category covers modifications related to adding, modifying or defining tools to automate code review associated task in the new workflow. We found 7 samples of this category (e.g., Figure 6 [24]).

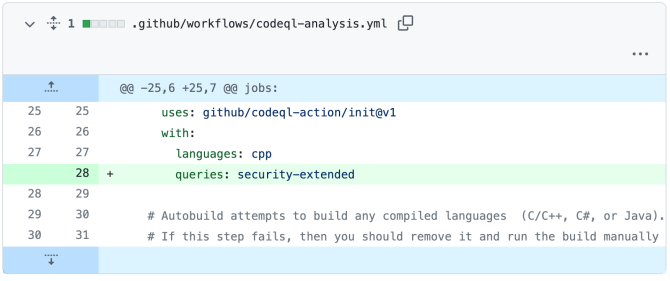

Fig. 6: Configuring a code review workflow [24].

Commented code (2). This category refers to code-comments present in the workflow. We found 2 samples of this category (e.g., [25]).

Code indentation (10). Code indentation refers to modifying the indentation of the code in the workflow. We found 10 samples of this category (e.g., [26]).

Testing configuration (16). Testing configuration refers to adding, modifying tools/software to test code in the workflow. We found 16 samples of this category (e.g., [27]).

Discussion. Many of the modifications related to Workflow Construction concern the way in which developers define and configure GA. Not surprisingly, the Instruction Definition category is our most frequent modification type (60.36%). In this case, the definition of a new instruction can be both involved in debugging [23] and adding features [28].

In this context, we notice lack of code documentation in workflow construction. For example, we find that just 17 samples of workflows that include code-comments.

V. Threats to Validity

We report construct, internal and external threats to validity.

Construct Validity: Threats to construct validity are related to the potential measurement imprecision when extracting data used in our study. Since we manually verified each of 222 workflow modifications, we did not discard entries in our dataset. In this way, we make sure that the automatic mining of workflows modifications do not include false positives.

Internal Validity: Threats to internal validity refers to confounding factors in our study that can affect the results because suggestiveness was introduced during the manual analysis. We mitigated this threat by following description criteria and by analyzing each sample (3 participants) independently.

External Validity: Threats to external validity are related to the ability to generalize our study observations. Although we considered different software projects in our analysis, our workflow modification taxonomy relies on the specific set of modifications that we analyzed and a small subset of available GitHub software projects. So, it is possible that in other contexts (other software projects), workflows modification types that we found did not appear.

VI. Related Work

Because GitHub Actions are a relatively new tool, released at the end of 2019, there is little work related to mining GitHub Actions workflows. Also, previous work has investigated other automation practices and tools such as continuous integration, continuous delivery and software bots.

GitHub Actions. Prior work about GitHub Actions technology has shown been well perceived by practitioners [3]. Its adoption shows an increasing number of monthly rejected pull requests and a decreasing number of commits on merged pull requests. The most common operations are continuous integration, miscellaneous utilities (e.g., reading configuration files), deployment, publishing, and code quality/code review, with other types of actions.

Continuous Integration and Continuous Delivery. The main goal of continuous integration and delivery is improving software quality and reducing risks and holds vast potential to further reduce human effort by automating repetitive tasks [29]. Nevertheless, this practice implies a cultural shift and involves non-trivial challenges. Hilton et al. [22] list these challenges as follows: (1) debugging build failures is not straightforward

since it is carried out on remote servers; (2) waiting for the results of long builds is necessary, which threatens developer productivity; (3) automating software engineering task requires dedicated developers; (4) the size of each coding task has to be granular enough to enable developers to integrate frequently their code contributions.

Software Bots. Software bots can be used for software engineering task automation. For example, in GitHub integrating software bots into the pull request workflow can perform tasks related to repairing bugs or refactoring the source code [30]–[32]. Also, software bots can support technical and social aspects of software development activities, such as communication (greeting messages) and decision-making. [33], [34].

VII. Conclusions and Future Work

We study workflow modifications from 10 different software projects. We qualitatively discussed our findings and expose implications for developers. Our effort contribute to (i) understanding how developers use GitHub Actions and (ii) providing a ground for possible future improvements.

Given the undeniable value of GitHub Actions technology we consider that the use tools is a need to better supporting workflows creation/edition. Tools should be able of identify syntax errors, and provide recommendations to specific common task. In addition, the workflow creation/editing process requires practitioners to accept and incorporate new approaches and culture to guaranty expected benefits of automation such as faster releases and increased developer productivity.

Our future work comprises the correlation between workflow modification and the workflow execution. Such a correlation would then be used to predict the build outcome without executing the workflow, and as such, making the workflow execution more agile. Another effort will comprise the build of IDE extension to support the refactoring and lint-like checks of GitHub Actions workflows. As far as we are aware of, the VSCode plugins related to workflow available in the Visual Studio Marketplace do not support refactoring and lint-like rules.

Acknowledgements. Bergel thanks the financial support of the ANID Fondecyt Regular project number 1200067. Valenzuela thanks the financial support of the Universidad de La Frontera, DIUFRO Project number DI20-0015.

References

[1] J. Humble and D. Farley, "Continuous delivery: Reliable software releases through build," *Test, and deployment automation. Pearson Education*, vol. 1, 2010.

[2] D. Kavaler, A. Trockman, B. Vasilescu, and V. Filkov, "Tool choice matters: Javascript quality assurance tools and usage outcomes in github projects," in *ICSE '19*.

[3] T. Kinsman, M. Wessel, M. A. Gerosa, and C. Treude, "How do software developers use github actions to automate their workflows?" *arXiv preprint arXiv:2103.12224*, 2021.

[4] N. Munaiah, S. Kroh, C. Cabrey, and M. Nagappan, "Curating github for engineered software projects," *Empirical Software Engineering*, vol. 22, no. 6, pp. 3219–3253, 2017.

[5] P. Valenzuela-Toledo and A. Bergel, "Workflow modifications dataset," July, 2021. [Online]. Available: https://bit.ly/3Dus7yL

[6] S. Few, "The encyclopedia of human-computer interaction," *The Encyclopedia of Human-Com-puter Interaction (p. Ch 35). Retrieved from https://www. interaction-design. org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/data-visualization-for-humanperception*, 2013.

[7] T. Zimmermann, "Card-sorting: From text to themes," in *Perspectives on data science for software engineering*. Elsevier, 2016, pp. 137–141.

[8] J. Saldaña, *The coding manual for qualitative researchers*. sage, 2013.

[9] Moosetechnology, "New workflow category commit example," July, 2021. [Online]. Available: https://bit.ly/3nNOxEF

[10] Django, "Deleted workflow category commit example," Github, July, 2021. [Online]. Available: https://bit.ly/3bz2RLs

[11] Roassal3, "Workflow file modification category commit example," Github, July, 2021. [Online]. Available: https://bit.ly/3byU6RQ

[12] Matplotlib, "New workflow category commit example," July, 2021. [Online]. Available: https://bit.ly/3CKYucs

[13] ——, "New workflow category commit example," July, 2021. [Online]. Available: https://bit.ly/2Y79ul6

[14] ——, "New workflow category commit example," July, 2021. [Online]. Available: https://bit.ly/2Y79ul6

[15] Django, "Deleted workflow category commit example," July, 2021. [Online]. Available: https://bit.ly/3nVWx6N

[16] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig, "Usage, costs, and benefits of continuous integration in open-source projects," in *ASE '16*.

[17] D. A. Tamburri, R. Kazman, and H. Fahimi, "The architect's role in community shepherding," *IEEE Software*, vol. 33, no. 6, pp. 70–79, 2016.

[18] Roassal3, "Showing instructions execution category commit example," July, 2021. [Online]. Available: https://bit.ly/3Fpi8eW

[19] Django, "Updated instructions category commit example," July, 2021. [Online]. Available: https://bit.ly/3kht8TB

[20] J. Ressia, A. Bergel, and O. Nierstrasz, "Object-centric debugging," in *ICSE '12*.

[21] Django, "Updated instructions category commit example," July, 2021. [Online]. Available: https://bit.ly/3GJq4co

[22] M. Hilton, N. Nelson, T. Tunnell, D. Marinov, and D. Dig, "Trade-offs in continuous integration: assurance, security, and flexibility," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*.

[23] Matplotlib, "Instruction definition category commit example," July, 2021. [Online]. Available: https://bit.ly/3GDXQzE

[24] Curl, "Code review configuration category commit example," July, 2021. [Online]. Available: https://bit.ly/3CO4W20

[25] Matplotlib, "Commented code category commit example," July, 2021. [Online]. Available: https://bit.ly/3jXeq3N

[26] Django, "Code indentation category commit example," July, 2021. [Online]. Available: https://bit.ly/3GBDM0V

[27] Curl, "Testing configuration category commit example," July, 2021. [Online]. Available: https://bit.ly/3weLFog

[28] Matplotlib, "Instruction definition category commit example," July, 2021. [Online]. Available: https://bit.ly/3nOSKYH

[29] P. M. Duvall, S. Matyas, and A. Glover, *Continuous integration: improving software quality and reducing risk*. Pearson Education, 2007.

[30] L. Erlenhov, F. G. de Oliveira Neto, R. Scandariato, and P. Leitner, "Current and future bots in software development," in *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*.

[31] M. Monperrus, "Explainable software bot contributions: Case study of automated bug fixes," in *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*. IEEE, 2019, pp. 12–15.

[32] M. Wyrich and J. Bogner, "Towards an autonomous bot for automatic source code refactoring," in *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*. IEEE, 2019, pp. 24–28.

[33] B. Lin, A. Zagalsky, M.-A. Storey, and A. Serebrenik, "Why developers are slacking off: Understanding how software teams use slack," in *Proceedings of the 19th acm conference on computer supported cooperative work and social computing companion*, 2016, pp. 333–336.

[34] M.-A. Storey and A. Zagalsky, "Disrupting developer productivity one bot at a time," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 928–931.