Exploring GitHub Actions through EGAD: An Experience Report

Pablo Valenzuela-Toledo[1,2], Alexandre Bergel[3], Timo Kehrer[1] and Oscar Nierstrasz[4]

[1]*University of Bern, Bern, Switzerland*
[2]*Universidad de La Frontera, Temuco, Chile*
[3]*RelationalAI, Bern, Switzerland*
[4]*feenk GmbH, Wabern, Switzerland*

Abstract

GitHub Actions (GA) is an automation and workflow orchestration platform that facilitates the execution of software engineering tasks. GA supports automation through YAML workflow files, the main component for action integration. The workflow files are a valuable source of information, for example, to understand how automation actions have been orchestrated. To conduct GA studies, researchers use batch (non-interactive) tools for analysis. However, this approach ignores the context of the GA domain and hinders understanding of the intricate pieces that revolve around workflow files.

We present our experience developing and using EGAD (Explorable GitHub Action Domain Model), a moldable domain-specific tool implemented on Glamorous Toolkit (GT) to depict and analyze detailed GA workflow data. We share valuable insights focusing on three key areas: (i) composing stories to develop domain models, (ii) conducting research by navigating custom views, and (iii) supporting the onboarding of researchers on GT. To facilitate takeaways from these lessons, we provide details that showcase our learnings, which researchers can apply in future endeavors, including investigating GitHub Actions.

Keywords

GitHub Actions, moldable development, software evolution

1. Introduction

Software development projects increasingly use GitHub Actions (GA) to automate non-trivial software engineering tasks [1]. Although GA was publicly released only in November 2019, it is already the dominant continuous integration service on GitHub [2]. With GitHub Actions, it is possible to automate actions depending on various triggers (such as commits, pull requests, issues, comments, *etc.*) and share them from one repository to another. This simplifies how developers build, test, and deploy software projects [3].

GA supports automation through workflows (YAML files) that describe the actions to be triggered by specific events. The workflow files represent a valuable source of information that

IWST 2023: International Workshop on Smalltalk Technologies. Lyon, France; August 29th-31st, 2023
*Corresponding author.
https://seg.inf.unibe.ch/people/pablo/ (P. Valenzuela-Toledo); https://bergel.eu (A. Bergel); https://seg.inf.unibe.ch/people/timo/ (T. Kehrer); https://www.oscar.nierstrasz.org (O. Nierstrasz)
0000-0001-6349-4296 (P. Valenzuela-Toledo); 0000-0001-8087-1903 (A. Bergel); 0000-0002-2582-5557 (T. Kehrer); 0000-0002-9975-9791 (O. Nierstrasz)

can be used, for example, to anticipate the effects of adopting such technology, or to understand how developers use it to improve daily software development activities [1, 3]. To take advantage of such data, researchers conduct studies into GA using batch (non-interactive) tools at the raw data level [1, 3, 4]. For example, Kinsman *et al.* [3] used Python scripts to analyse GA adoption trends and patterns. However, although this approach is widely applicable, it ignores the contextual nature of a study and does not provide domain objects, source code, or narratives that link them [5]. This is a problem since, as Nierstrasz and Girba showed [6], domain-specific knowledge can ease answering questions about software systems.

To overcome this limitation, we introduced the **E**xplorable **G**itHub **A**ction **D**omain Model (EGAD), a domain-specific tool to depict and analyze GA workflows and their evolution [7]. EGAD provides (i) an explorable domain model, (ii) custom views, and (iii) live narratives that access the raw GA workflow data.

In this paper, we report on our experience developing EGAD and its use as a research workbench. First, we briefly discuss the need for developing this tool. Then, we reflect on three main lessons we learned from:

1. Composing stories to develop domain models,
2. Conducting research by navigating custom views, and
3. Supporting the onboarding of researchers on Glamorous Toolkit[1].

We document these lessons so practitioners and researchers can use them in future efforts.

In the next section, we explain the need for an explorable GA domain model. In Section 3, we present EGAD. In Section 4, we present lessons learned in the development of EGAD. In Section 5, we discuss shortcomings and open challenges for future work. We conclude with a few closing remarks in Section 6.

2. The need for an explorable GitHub Action domain model

Because GA is a relatively new platform released at the end of 2019, there exists little work investigating GA workflows.

Kinsman *et al.* [3] quantitatively analyzed the impact of adopting GA in around 3,000 repositories. They show that adopting GA in software repositories increases the number of rejected pull requests and decreases the number of commits in merged pull requests. In addition, by manually inspecting related GA issues, they found that developers have a positive perception of this platform. These results are especially relevant for practitioners to understand and prevent adverse effects in GA adoption.

Valenzuela-Toledo and Bergel [4] investigated the versioning practices followed by actions and the workflows relying on them. They conducted an empirical study inspecting 222 commits of GitHub Actions workflows obtained from 10 popular open-source repositories. As a result, they categorize and tag workflow modifications, identifying 11 types of changes and revealing opportunities for improvement in how workflows are built and edited. In particular, these results highlight the need for adequate tooling to support refactoring, debugging, and code editing of GA workflows.

[1] https://gtoolkit.com

Similarly, Decan *et al.* [1] investigated trends and adoption patterns in the emerging GA ecosystem. In their study, they investigate the use of GitHub Actions on a dataset of 68K repositories on GitHub, of which 43.9% were using GitHub Actions workflows. They analyze how workflows implement automation and identify the most frequent practices. They show that reusing actions is a common practice, and about half of all the workflow steps rely on reusable actions, mostly originating from public repositories. They also study the most frequently used actions and how workflows refer to them. For example, they verify that most actions are used mainly for development purposes and are mostly triggered by push or pull request events.

Previous studies have provided access to source code and related datasets, but have been limited in their perspective. For instance, Decan *et al.* [1] extracted actions from each workflow in their dataset, and created dataframes[2] to prioritize the data structure over the main subject of the domain: the GA. As a result, the GA workflow is not the core concern of the research domain, thus not being considered a first-class citizen. This pattern is consistently observed in the other previously revised works.

However, the natural context of GA is broad and rich in features, including code specifications, workflow versioning, and tool dependencies. Consequently, to thoroughly investigate the natural context of GA, it is necessary to consider the main features of GA workflows and an *Explainable System* approach. In this approach, a system is explainable if it is easy to create narratives that link documentation, code, and live objects [6]. This is enabled by creating a live and explorable domain model as a tool able to support GA empirical research and provide features to explain complex software systems.

EGAD shifts the focus from previous studies by providing an explorable, reusable, and extensible Domain Model to research the GA ecosystem.

3. EGAD

To support researchers in studying GA workflows, EGAD has been developed as a domain-specific tool that provides a comprehensive domain model, customizable views, and narratives, enabling easy exploration and analysis of GA workflows.

The architecture of EGAD consists of (i) an explorable domain model, (ii) custom views, and (iii) live narratives that access the raw repository data [7].

The domain model wraps the GA workflow data. The model includes: (i) the workflow history, including all the commits associated with a GA workflow, and (ii) the representation of the workflow, including events, jobs and steps.

A class diagram focusing on the most important entities is shown in Figure 1. A `WEHistories` instance contains a collection of `WEHistory` objects, each of which represents the history of workflows of a dedicated project. A `WEFileCommit` object represents a commit revising a dedicated workflow by a new version (`WEWorkflow`), which in turn consists of events, jobs and steps.

Custom views enable the exploration and navigation of the domain model from multiple perspectives. Instead of presenting the data generically, custom views provide critical insights

[2] https://pandas.pydata.org/docs/reference/frame.html

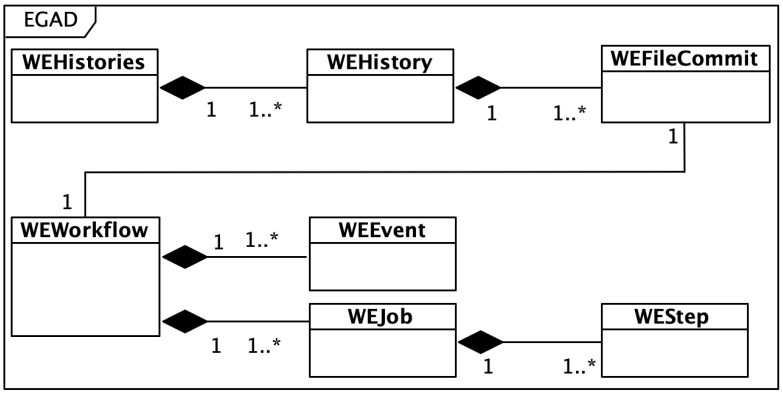

Figure 1: Representation of GA workflows and their evolution.

into the domain model, and make it possible to pose questions and explore hypotheses. In order to answer domain-specific questions, the views can be easily extended.

Narratives link documentation, source code, and running objects to specific questions or hypotheses of the domain model. We use narratives to explain use cases, scenarios, and features of the GA domain.

The composition of a narrative is depicted in subsection 4.1, which showcases how documentation, source code, and examples are linked together.

The tool's full replication package, including documentation and an example dataset, is available for researchers to run the tool and conduct their analyses [7]. A stable version of EGAD is hosted on Zenodo for ease of access [8].

4. What have we learned in developing EGAD?

This section presents lessons learned from the development of EGAD in three important ways: (i) *research by navigating custom views*, (ii) *composing a story*, and (iii) *supporting the onboarding of researchers on GT*. Through these lessons, we offer insights into the challenges we faced in developing EGAD.

4.1. Compose a story

EGAD is being developed on top of the Glamorous Toolkit, a moldable environment for building live and explorable domain models using hyperlinked notebooks, live domain objects, and customizable, domain-specific views [6]. GT includes extensive live documentation detailing how to customize it for a given application domain. GT is built on Pharo,[3] a modern, open-source Smalltalk environment.

To develop EGAD, we adopted an Explainable System approach based on the ideas of Nierstrasz and Girba [6]. This involves incorporating key features of GT, such as Lepiter notebooks,

[3] https://pharo.org

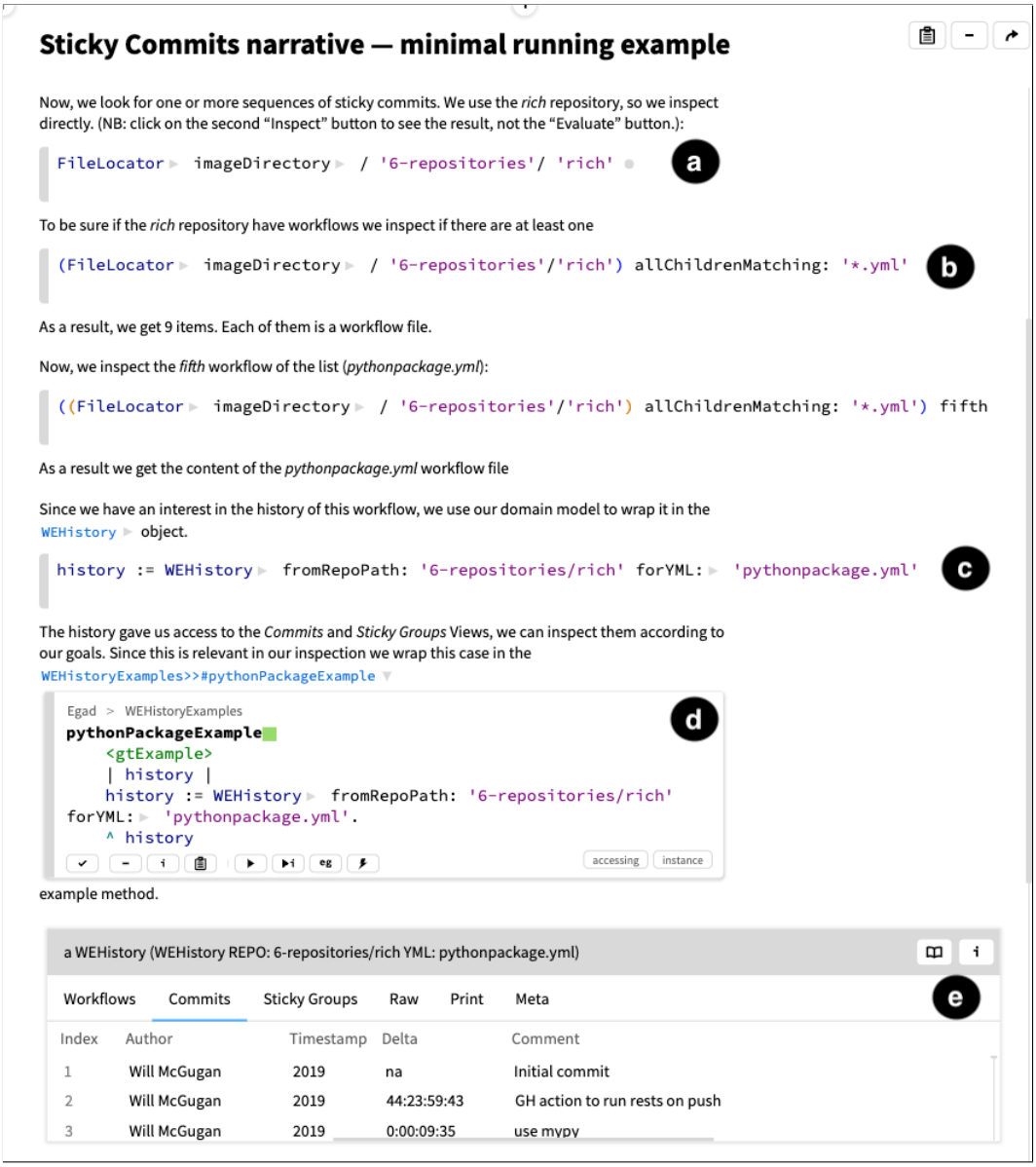

Figure 2: Composing a story. Using our GA domain model, we wrap a YAML workflow file and include them in a runnable example to assemble a story.

example methods (a particular object to test and wrap use cases), and a moldable inspector with custom views.

First, we leveraged Lepiter notebooks as a key element for the development of EGAD. This allowed us to create interactive, executable documentation that can be used to explore and understand the GA domain.

Figure 2 illustrates our approach. First, we inspect the `rich` repository (Figure 2a). Next, we identify all the YAML files available in the `/github/workflows` directory and inspect

the one named `pythonpackage.yml` (Figure 2b). In order to make a story, we wrap it using the domain model and turn it into an example (Figure 2c). This example is executable, and it can be inspected. In the narrative, the example has two parts. The first one explicitly shows the example source code (Figure 2d). The second shows the result of inspecting the example, presenting specific views of EGAD (Figure 2e). As shown in the figure, both parts of the example are presented in the same narrative.

This case illustrates how we *compose a story*, documenting our task and progress and linking documentation, source code, and running examples. By providing concrete examples to analyze and explore GA workflows, we were able to explain particular scenarios of interest. This feature is implemented to explore the boundaries of the GA domain and ensure that our domain model is comprehensive and effective. Moreover, custom views facilitate the identification of potential bottlenecks or inefficiencies in the workflow, allowing researchers to make informed decisions on how to optimize the workflow.

We continue to develop EGAD by taking an Explainable System approach to building the GA domain model, and leveraging key GT features such as Lepiter notebooks and example methods.

4.2. Research by navigating custom views

The evolution of GA is an area of interest for researchers as it can help developers to understand how to modify and adapt their workflows over time [4, 9]. The evolution of workflows can be influenced by various factors, such as changes in the development process, updates to tools and libraries, or new requirements. Researchers can use EGAD to analyze these changes and gain insights into how workflow specifications are modified over time.

The evolution of GA workflows presents interesting change patterns. Change patterns can reveal significant information regarding how developers modify their workflows, and what factors drive these changes. Figure 3 illustrates how we inspect and navigate GA data to investigate a particular pattern of change. We inspect the example described in subsection 4.1 (Figure 3a), and navigate it with the help of multiple custom views.

We access the `WEHistory` view that shows groups of commits (Figure 3b). These commits only relate to the YAML file, and are grouped by author name, with the time between commits being under sixty minutes. This view also shows the date of the first and last commits, and the total number of commits that belong to this group.

Next, we inspect the first element of the `WEHistory` view. As a result we access the `WEStickyGroup` view (Figure 3c). We create this view to inspect a series of continuous commits that were made by the same developer to correct errors in the workflow specifications made in the previous commit. These types of commits may suggest that the developer is struggling to implement certain GA behaviors correctly. The name “Sticky Group” was chosen as a metaphor to describe the continuous nature of these commits, as they seem to stick to one another. This view presents a list of all commits belonging to the same author. The view includes the commit date, the `Duration` (or delta time between commits), and the commit comments. All the commits listed in this view are inspectable. Here (Figure 3c), we inspect the commit with `index 22`. As a result, we get the `WEFileCommit` view, which includes `Details` of the objects, `Changes` of the related workflows files, and two optional diff views. In Figure 3d, we show the `Diff(two panes)` views to highlight the differences between commits.

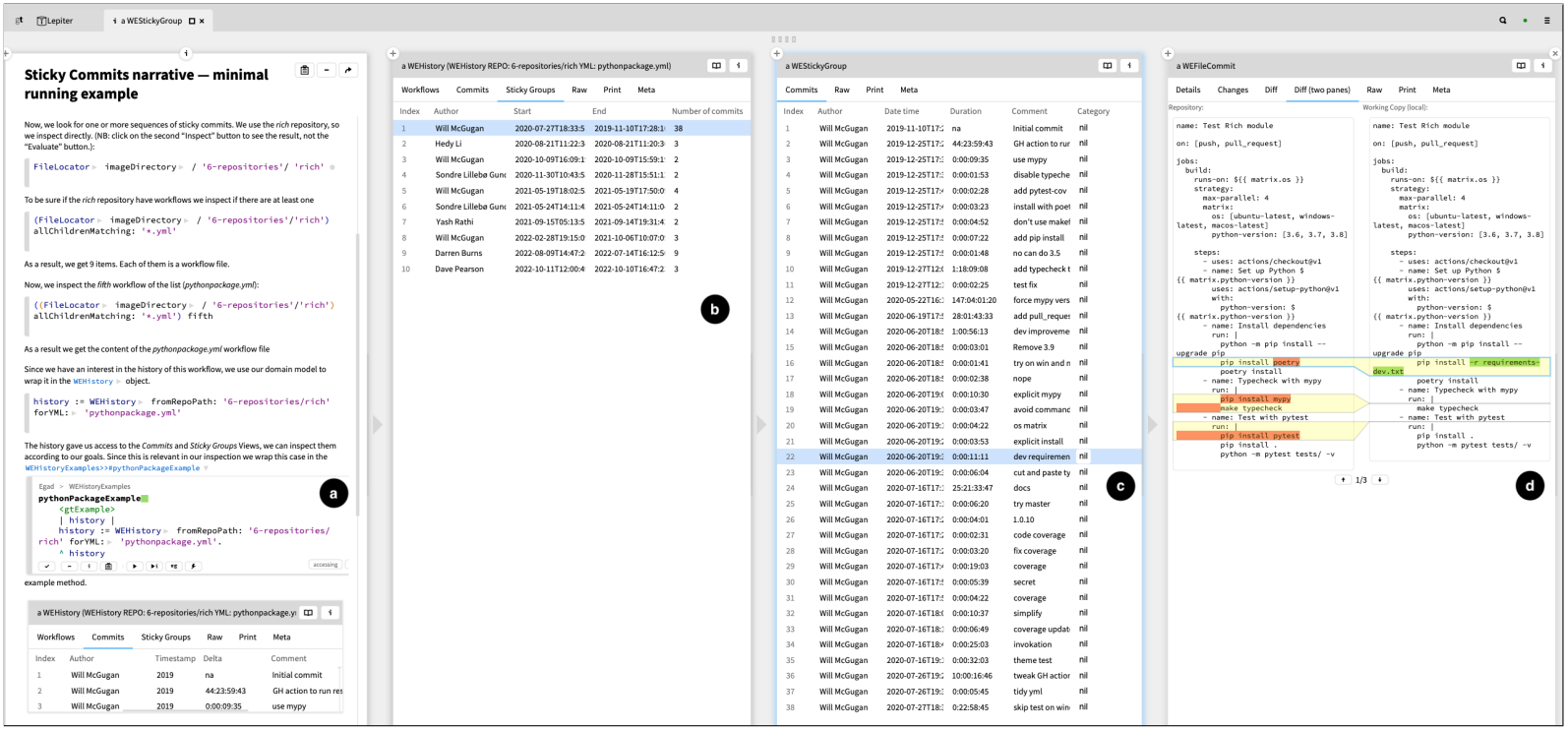

Figure 3: Illustration of how we conduct *research by navigating custom views* using EGAD.

The previous description shows how we conduct *research by navigating custom views* using EGAD. EGAD can be used as a research workbench that allows researchers to analyze and explore workflows in detail, uncovering valuable insights into the development process and identifying areas for improvement.

4.3. Support the onboarding of researchers on GT

Integrating GT technologies has enhanced our ability to conduct exploratory research. However, it is essential to acknowledge that leveraging these technologies entails a learning curve, particularly as their adoption cannot be assumed for every PhD student or researcher. This is why a comprehensive onboarding process becomes crucial.

Onboarding refers to the process of introducing software developers to a new project, its source code, and a new development team [10]. Unfortunately, this process is challenging and often time-consuming, and thus often considered a "necessary evil" in the software development industry [11, 12].

The software engineering research community has proposed various techniques to address onboarding barriers. Collaborative strategies, such as pair programming and mentoring, have been shown to enhance the learning process and accelerate the adaptation of new developers. Therefore, incorporating these approaches can be beneficial for organizations looking to improve their onboarding process and facilitate the integration of new employees into their teams [13, 14]. In addition, the research community has also put forward several remote onboarding recommendations that can facilitate the training and integration of new employees into virtual work environments [10]. Although there is no guarantee of a flawless onboarding experience, the aforementioned strategies do offer a systematic and proven approach to mitigating the challenges associated with the process. By implementing these techniques, organizations can reduce the time and effort required to onboard new employees and ensure a smoother integration into the team and project [10].

Our experience developing EGAD involved successfully onboarding a new developer (PhD student researcher) into the Glamorous Toolkit (GT) ecosystem, leveraging several techniques identified by Rodegher *et al.* [10]. One of the primary techniques we used was assigning an onboarding technical mentor, who provided guidance and support throughout the process, ensuring that the new developer had access to the necessary resources and information to navigate the technical challenges. This approach allowed for a more streamlined onboarding process, enabling the new developer to quickly integrate into the research environment and contribute to the project's success.

To further enhance the success of the onboarding process, we also scheduled regular 1:1 online meetings between the developer and their onboarding mentor. These meetings provided an opportunity for the developer to ask questions, share progress updates, and receive feedback. This approach allowed for a more personalized and tailored onboarding experience that addressed the developer's specific needs and concerns. The online format of the meetings also made it easier for the developer and mentor to attend and participate, regardless of their location or schedule constraints.

In addition, we encourage the use of GT's key features. First, we use as a documentation resource the GT book, which provides a shared knowledge base containing relevant articles,

tutorials, and examples. Second, we document tasks and progress using Lepiter notebooks. By utilizing this feature, we facilitate the comprehension of developing the GA domain model. Figure 2 illustrates the documentation of the progress when the new developer was implementing the GA domain model. Lastly, we take advantage of the system's explorable design to better understand the project's technology stack.

Our approach to onboarding the developer into the EGAD project was successful, primarily due to its multifaceted nature. One key aspect of the onboarding process was the use of an onboarding mentor who provided invaluable guidance and support to the new developer. Additionally, scheduling regular 1:1 online meetings was also a critical factor in the success of our approach, in spite of time zone and agenda issues. We faced some challenges during the onboarding process, such as the developer having difficulty navigating the project's technical complexities, despite our efforts to provide clear documentation and accessible resources. Nevertheless, our experience shows that our onboarding approach proved effective, and we successfully integrated the new developer using GT technologies.

4.4. Summary

We reflected on the development of EGAD and shared three key lessons we have learned. A summary of these lessons is presented below:

- *Compose a story*: Do you need to explain step by step how you developed your tool? Then compose a story. It is possible by adopting an Explainable System approach, incorporating key features of GT, such as Lepiter notebooks, example methods, and a moldable inspector with custom views.
- *Research by navigating custom views*: Do you want to see your investigation step by step? Instead of using different tools to develop your GA research, use EGAD to navigate and explore the GA domain model in detail. As a result, you can uncover valuable insights into the development process and identify areas for improvement.
- *Supporting the onboarding of researchers on GT:* Do you want to succeed in onboarding new researchers? Use a comprehensive approach to onboard newcomers into the GT ecosystem. Assign a mentor, schedule regular online meetings, and encourage using GT key resources like Lepiter notebooks, the explorable design, and the GT book to succeed in this process.

5. Limitations

Our experience report describes lessons learned while developing EGAD. However, these lessons are limited to our context, which may affect the validity and generalization of our observations.

Our lessons draw on our research work, which carries the inherent risk of subjectivity and limits the generalization of our findings to other contexts. To address this threat, we have described our experiences in detail and framed them within the context of the Explainable System approach implemented in GT. This approach provides a systematic and well-defined framework for developing and evaluating complex software systems, which increases the likelihood that our experiences will be relevant and valuable to other researchers and practitioners. Despite

this, we acknowledge that the unique characteristics of our project and context may limit the applicability of our insights to different settings, and further research is needed to validate and extend our findings.

In addition to this limitation, there are also open issues that need to be addressed in EGAD. One such issue is the validation of the tool. While EGAD has been used for research, there has not yet been a comprehensive validation of the tool's effectiveness. This means that it is difficult to assess the accuracy and reliability of the insights generated by EGAD. To address this issue, researchers could conduct studies that compare the insights generated by EGAD to those generated by other tools or manual analysis.

Overall, while these lessons represent valuable insights for analyzing and exploring GitHub Actions workflows, researchers should be aware of their limitations. By addressing these issues, researchers can ensure that our recommendations remain a reliable experience in the future.

6. Conclusion

To sketch out how we have conducted GA research, we have reported our experience developing EGAD, a tool for inspecting and navigating GA workflow data. EGAD was developed to conduct research by adopting an Explainable System approach, which means we took advantage of key features of GT, such as Lepiter notebooks, example methods, and custom views.

The key lessons regarding GA research that we draw are: (i) *Research by navigating custom views* – using EGAD to navigate and explore the GA domain model in detail can help to uncover valuable insights into the development process and identify areas for improvement, (ii) *Compose a story* – adopting an Explainable System approach using key features of GT, such as Lepiter notebooks, example methods, and custom views, can make each step of your development and research process explainable, and (iii) *Support the onboarding of researchers on GT* – a comprehensive approach that includes a mentor, regular meetings, and GT key resources can lead to success in this process.

While our experience developing and using EGAD has provided valuable insights into the potential of change-enabled software systems, there are still many avenues for future work to explore. Some potential directions include: (i) expanding the domain model to include other entities that directly interact with GA (for example, issues and pull requests), and (ii) validating the tool's effectiveness.

References

[1] A. Decan, T. Mens, P. R. Mazrae, M. Golzadeh, On the use of GitHub actions in software development repositories, in: 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, 2022, pp. 235–245.

[2] M. Golzadeh, A. Decan, T. Mens, On the rise and fall of CI services in GitHub, in: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 2022, pp. 662–672.

[3] T. Kinsman, M. Wessel, M. A. Gerosa, C. Treude, How do software developers use GitHub

actions to automate their workflows?, in: 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), IEEE, 2021, pp. 420–431.
[4] P. Valenzuela-Toledo, A. Bergel, Evolution of GitHub action workflows, in: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 2022, pp. 123–127.
[5] A. Chiş, T. Gîrba, J. Kubelka, O. Nierstrasz, S. Reichhart, A. Syrel, Moldable tools for object-oriented development, in: Present and Ulterior Software Engineering, Springer, 2017, pp. 77–101.
[6] O. Nierstrasz, T. Gîrba, Making systems explainable, in: 2022 Working Conference on Software Visualization (VISSOFT), IEEE, 2022, pp. 1–4.
[7] P. Valenzuela-Toledo, A. Bergel, T. Kehrer, O. Nierstrasz, EGAD: A moldable tool for GitHub Action analysis, in: To be published in IEEE/ACM 20th International Conference on Mining Software Repositories (MSR), IEEE, 2023, pp. –.
[8] P. Valenzuela-Toledo, A. Bergel, T. Kehrer, O. Nierstrasz, EGAD: A moldable tool for GitHub Action analysis, 2023. URL: https://doi.org/10.5281/zenodo.7714219. doi:10.5281/zenodo.7714219, tool repository: https://github.com/pavt/egad.
[9] P. Valenzuela-Toledo, A. Bergel, T. Kehrer, O. Nierstrasz, EGAD: a Moldable Tool for GitHub Action Analysis, 2023. URL: https://github.com/pavt/egad. doi:10.5281/zenodo.7714219.
[10] P. Rodeghero, T. Zimmermann, B. Houck, D. Ford, Please turn your cameras on: Remote onboarding of software developers during a pandemic, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), IEEE, 2021, pp. 41–50.
[11] A. Labuschagne, R. Holmes, Do onboarding programs work?, in: 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, IEEE, 2015, pp. 381–385.
[12] J. Bonar, E. Soloway, Uncovering principles of novice programming, in: Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, 1983, pp. 10–13.
[13] L. Williams, A. Shukla, A. I. Antón, An initial exploration of the relationship between pair programming and Brooks' law, in: Agile Development Conference, IEEE, 2004, pp. 11–20.
[14] M. Johnson, M. Senges, Learning to be a programmer in a complex organization: A case study on practice-based learning during the onboarding process at Google, Journal of Workplace Learning (2010).