From Specialized Mechanics to Project Butlers

The Usage of Bots in Open Source Software Development

Zhendong Wang, University of California, Irvine
Yi Wang, Beijing, University of Posts and Telecommunications
David Redmiles, University of California, Irvine

// We seek to identify how open source software (OSS) projects adopt bot services from a diverse set of selections. Our empirical research examines bot usage in the most popular OSS repositories in GitHub. //

TODAY, MAINSTREAM OPEN source software (OSS) development often involves a huge base of contributors, users, and other stakeholders, which is afforded by various online social coding platforms, e.g., GitHub. At the same time, the fast, ongoing penetration of the continuous integration, delivery, and deployment (CI/CD) model makes urgent demands on rapid iterations. Facing increasing pressures from managing a large number of contributors and their contributions and the ever-faster pace in CI/CD, some small-scale automation techniques have found their way into OSS projects’ engineering practices.

Software agents for software engineering (SE) tasks, i.e., developer bots, or bots, for short, are exactly such automation techniques that have the potential to significantly automate many aspects of building software. Various bots have been proposed for improving developers’ workflow through better productivity and work–life balance. These software agents and bots automate repetitive and routine tasks, such as managing issues and pull requests, building and executing test suites, and so on. Thus, they can often substitute for human labor. While these bots have saved substantial human efforts, they also have exhibited limitations, for instance, not being designed for smart tasks and lacking interactivity during human–bot communication.

Bots will inevitably be adopted by more and more OSS projects in the foreseeable future of automated SE. Thanks to the continuous efforts of practitioners and researchers, the current bot market offers a wide range of development services for OSS maintainers to shop around. However, we have little knowledge about how practitioners select and adopt these bots.

Here, we start to bridge the gap by empirically examining the bots adopted by popular OSS projects on GitHub, the largest online social coding platform. Combining automated bot detection and manual review, we compile a list of bots used in 1,000 popular projects. Analyses of these bot services show a remarkable adoption rate (61%) among popular GitHub OSS projects when compared with prior studies. However, the
application scenarios of these bots are still limited: many of them are combinations of simple automation features, which are triggered by specific project events, and they perform designated tasks through a rule-based mechanism. In addition, over 69% of bots remain private, without transparently sharing their implementation and being accessible outside projects and organizational ecosystems.

Bots in OSS Development

In the past decade, the emerging CI/CD model and the large-scale collaborative software development powered by online social coding platforms have driven the rapid evolution of OSS development practices. Fast iterations drastically complicate the development process as projects evolve. Therefore, practitioners resolve to exploit automation techniques that handle specific routine development and social tasks so that they can focus on innovative activities.

These automation techniques, i.e., bots, may support developers through many service mediums. Some bots have user profiles and are authorized as their projects’ contributors. Besides these account-based bots, others may employ platform applications, user settings, and various external services. Practitioners often adopt desired bot services with tradeoffs among performance, privacy, and security.

SE practitioners and researchers have put significant effort into bot development. Several bot creation frameworks have been developed to separate bot operation logic from specific platform mechanisms, such as GitHub’s WebHook event subscription. For example, Probot allows bot developers to interact with GitHub via an internal event emitter, and therefore, developers may focus on designing bot logic. There are other bot creation frameworks that bring specific engineering benefits, such as OpenBot, Fabric Bot, and Windows Community Tool.

Meanwhile, researchers have collected practitioners’ feedback for future improvement. For instance, they find that practitioners expect more smart features in bots’ communication and functionality. Erlenhov et al. also emphasize the importance for a bot to be socially competent and bring technical value, especially when it is designed for smarter and more complex tasks.

Liu et al.’s survey created seven principles for designing bots that demonstrate the critical challenges in interactions.

Methods

This study extracted activity traces of bot accounts and apps from GitHub repositories and identified bots through automatic bot detection and manual review. The data we sought included the following:

- a list of popular software development repositories that have employed bots in CI/CD
- descriptions of bot accounts, apps, and services employed by these repositories.

Sampled Repositories

The number of GitHub stars is an indicator of popularity and is also associated with project evolution. Sampling most starred repositories helps to avoid many perils in mining repositories, e.g., projects that received minimal participation. Moreover, we ensured that the sample included only software development projects using the pull request model, which excluded non-SE repositories. Finally, due to the limitations of manual reviewing, e.g., substantial time effort, the sample size was set as 1,000.

Data Collection and Analysis

First, to automatically identify bot activities from GitHub repositories, we adopted a bot identification method called BODEGHA, a machine learning classifier using language patterns and features of comments to detect bot activities. Second, we manually examined the automatically classified bot accounts/services and reviewed their activities for data reliability.

We collected activity data from the most recent list of event actors in commits, issues, and pull requests. We also reviewed repository information documents, e.g., readme.md. When we found an event actor that was suspiciously repeating similar activities or demonstrated that the content was automatically generated, we first reviewed its GitHub profile or GitHub Marketplace/app listings for its naming and description. Second, we reviewed the content of this GitHub account’s recent comments and activities. Finally, for some accounts that did not display their contributions, we collected their activity records through the GitHub application programming interface.

Two researchers generated the preceding review protocol for examining and identifying bot services and tested it on a random subsample of 100 repositories. The process achieved a 0.645 interrater reliability (Cohen’s kappa), suggesting substantial agreement. Any differences were resolved during a discussion session and led to updating the protocol. Finally, three researchers followed the updated protocol and labeled the entire sample.

For each identified bot, we extracted several data items, including the bot’s service-providing medium, main functionality, service and source code availability, and design/creation framework, if indicated. In addition, we collected the external reference of
the bot, including its documentation site, GitHub profile page, and main service medium site. Some sets of bots providing the same service often under one application, such as Codecov IO and Codecov Commenter and Dependabot Preview and Dependabot, were merged into one service. A replication package of the manual review protocol and bot data is available at https://tinyurl.com/33392bp3.

**Results**

Among the 1,000 sampled repositories, BODEGHA identified bot activities in 462. Following the steps described in the “Data Collection and Analysis” section, our manual review found that 613 sampled repositories employed 201 distinctive bot services, though three of them were reported as deprecated or stopped services.

Based on our data, bots’ functions fall into six categories:

- **CI task assistance**: The service automates the various tasks for CI/CD, including managing branches and releases (Release Drafter and Semantic Release Bot), pushing commits and merging approved/passed pull requests (Bors, LGTM, and Repo Ranger), and automating building, testing, and deployment (Vercel, Buildbot, and Bors).
- **Issue and pull request management**: The service automates management of issue and pull request tracking systems, including labeling issues and pull requests, cleaning inactive/invalid requests (Mary Poppins, Carsonbot, and Support), and requesting and formatting issue and pull request content through checklists (Issue Check, Request Info, and Vue Bot).
- **Code review assistance**: The service provides assistance in the code review process, including performing static and dynamic analysis (Sourcery AI, SourceLevel Bot, and Codecov), change-set summaries and difference visualization (Changest Bot, ECMA262 Compare Bot, and Sizebot), and code reviewer assignment (Googlebot, PullApprove, and VS Code Triage Bot).
- **Dependency and security**: The service periodically checks and ensures that a repository’s dependency is update to date, and it detects and reports security issues (Dependabot, Depfu, and Renovate).
- **Developer and user community support**: The service assists with the management of a community through collecting contributor license agreements, acknowledging contributors (All Contributors), managing contributor permissions (MeeseeksDev), and automatically commenting for welcoming contributors and providing feedback (Mirai Bot and Welcome).
- **Documentation generation**: The service assists in generating end user and developer documentation (DecDocs Bot, Weekly Digest, and RC Publisher).

These application scenarios and functions of adopted bot services are similar to the findings in prior research. Additionally, in our research, we examine combinations of bot functions. Most prevalent is the sole function for automating CI tasks. Combinations of functions are less prevalent in terms of distinctive bot services (refer to Figure 1 for more details).
| Bot service | Service medium | Interactions | Cost for developers | Functionalities | Code availability | Service availability | Creation framework | Number of projects |
|-------------|----------------|--------------|---------------------|-----------------|------------------|---------------------|-------------------|------------------|
| Dependabot  | GitHub repository security settings | Pull request and pull request comment | Free | Update dependencies of a repository from automatically generated pull requests | Yes | Public | NA | 171 |
| Stale       | GitHub apps | Issue and pull request comments | Free | Label inactive issues and pull requests as “stale” and close them if they remain inactive | Yes | Public | ProBot | 100 |
|Codecov      | GitHub apps and actions | Pull request comments | Free and paid | Report the test coverage of a pull request | No | Public | NA | 89 |
|Googlebot    | GitHub apps authorized account | Pull request comments | Private | Check and collect contributor license agreements (CLAs) and label an issue or pull request with its CLA status. Assign developers to issue and pull requests. Collect user feedback after an issue closes | Yes | Organizational | ProBot | 51 |
|CLA Assistant| GitHub apps and actions | Pull request comments | Free | Comments on pull requests to ask contributors to sign CLAs | Yes | Public | NA | 46 |
|Renovate     | GitHub apps | Pull request and pull request comments | Free | Update dependencies of a repository from automatically generated pull requests | Yes | Public | NA | 34 |
|Facebook Github Bot | Authorized account | Pull request comments | Private | Comment on a pull request to ask the contributor to sign a CLA. Make commit and push changes. Summarize changes to a pull request and execute automatic tests. Label the CI status on a pull request | No | Organizational | NA | 30 |
|Coveralls    | GitHub apps and actions | Pull request comments | Free and paid | Report the test coverage of a pull request | No | Public | NA | 26 |
|Vercel       | GitHub apps and actions | Pull request comments | Free and paid | Automate CI for building, testing, and deployment for front-end applications. Comments on pull requests include URL of a preview of deployed changes | No | Public | NA | 20 |

*We do not include GitHub Action as a typical bot service in this study because it is usually powered by existing as well as customized automation.*
Many bots have specialized in multiple tasks and become a butler for a repository. For example, RepoKit-teh performs multiple tasks, including checking the format of a pull request, automating tests on a pull request, assigning users to issue and labeling problems, and merging pull requests that passed automatic tests. Some organizations have even customized a private butler bot overseeing their CI/CD practices, e.g., Googlebot and Facebook GitHub Bot in Table 1.

However, the popular bots we identified employed similar design mechanisms and provided limited interactivity. For a limited number of OS bots, the implementation was with a rule-based system that subscribed to and acted based on certain repository events and event payloads. For example, when offering automatic issue labeling, Carsonbot required a strictly formatted issue with content in its template. Moreover, the interactivity of the bots has not been improved, for various reasons. The basic mechanism is still to respond and act under pre-defined input for completing simple, tedious tasks. However, we observe the prevalence of multitasking butlers in popular GitHub repositories, and we anticipate a growing percentage of bots that automate multiple tasks instead of specializing in just one.

Finally, for over two-thirds (69%) of the identified bots, their implementations remained opaque to the public. Moreover, more than half (53%) were available and applicable only to specific repositories and organizations. We argue that these two major reasons for bots to be kept private and closed source, according to our observations.

Discussion
Over 60% of the sampled repositories employed at least one bot to automate routine workflows. The high adoption rate suggests that SE bots have become prevalent in OSS projects. Although bots have not been sophisticated enough to handle advanced tasks in practice, the convenience they provide has outweighed many of their drawbacks, e.g., disruptive notifications. The current practice of OSS development on GitHub has largely become a semiautomated procedure heavily assisted by bots.

Discussion
Over 60% of the sampled repositories employed at least one bot to automate routine workflows. The high adoption rate suggests that SE bots have become prevalent in OSS projects. Although bots have not been sophisticated enough to handle advanced tasks in practice, the convenience they provide has outweighed many of their drawbacks, e.g., disruptive notifications. The current practice of OSS development on GitHub has largely become a semiautomated procedure heavily assisted by bots.

Finally, we list the most prevalent bot services, based on how many projects they appear in, in Table 1, and we spot a few common patterns for their services. Noticeably, these services expect Facebook GitHub Bot to provide a quick setup option with GitHub apps and actions. Pull request comments are the major communication channel for them, in addition to Stale, which is also specialized with issues. Further, these services provide a free installment option to the public, though two services are subject only to their organizational ecosystems.

Bots will inevitably be adopted by more and more OSS projects in the foreseeable future of automated SE.

T

his study leveraged automated and manual reviews to investigate the adoption of bots in OSS development practice on GitHub. We found that bots had become more prevalent in OSS development, as expected. However, somewhat unexpectedly, SE bots had not evolved much beyond atomic application scenarios but simply integrated more functions and multiple tasks. This study challenged our ideas about the future of bots. Namely, one line of future work could be to provide an interactive interface for butler bots while communicating with developers during CI/CD practices. Another line of research could be to develop a mechanism or framework to enable a bot-sharing community and address the concerns of security.

Acknowledgments
We would like to thank the editor and anonymous reviewers. Zhendong Wang and David Redmiles are
supported, in part, by the Donald Bren School of Information and Computer Sciences, University of California, Irvine. Yi Wang is partially supported by the National Natural Science Foundation of China, under grant 62172049.

References

1. M. Gerosa et al., “The shifting sands of motivation: Revisiting what drives contributors in open source,” in Proc. IEEE 43rd Int. Conf. Softw. Eng., 2021, pp. 1046–1058, doi: 10.1109/ICSE43902.2021.00098.

2. A. N. Meyer, L. E. Barton, G. C. Murphy, T. Zimmermann, and T. Fritz, “The work life of developers: Activities, switches and perceived productivity,” IEEE Trans. Softw. Eng., vol. 43, no. 12, pp. 1178–1193, 2017, doi: 10.1109/TSE.2017.2656886.

3. D. Liu, M. J. Smith, and K. Veeramachaneni, “Understanding user-bot interactions for small-scale automation in open-source development,” in Proc. CHI Conf. Hum. Factors Comput. Syst., 2020, pp. 1–8, doi: 10.1145/3334480.3382998.

4. M. Wessel et al., “The power of bots: Characterizing and understanding bots in OSS projects,” Proc. ACM Hum.-Comput. Interact., vol. 2, no. CSCW, pp. 1–19, 2018, doi: 10.1145/3274451.

5. L. Erlenkov, F. G. de Oliveira Neto, R. Scandariato, and P. Leitner, “Current and future bots in software development,” in Proc. IEEE/ACM 1st Int. Workshop Bots Softw. Eng. (BotSE), 2019, pp. 7–11, doi: 10.1109/BotSE.2019.00009.

6. C. Lebeuf, M.-A. Storey, and A. Zagalsky, “Software bots,” IEEE Softw., vol. 35, no. 1, pp. 18–23, 2018, doi: 10.1109/MS.2017.4541027.

7. A. Abdellatif, D. Costa, K. Badran, R. Abdalkareem, and E. Shihab, “Challenges in chatbot development: A study of stack overflow posts,” in Proc. 17th Int. Conf. Mining Softw. Repositories, 2020, pp. 174–185, doi: 10.1145/3379597.3387472.

8. H. Borges and M. T. Valente, “What’s in a GitHub star? Understanding repository starrning practices in a social coding platform,” J. Syst. Softw., vol. 146, pp. 112–129, Dec. 2018, doi: 10.1016/j.jss.2018.09.016.

9. E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, “The promises and perils of mining GitHub,” in Proc. 11th Working Conf. Mining Softw. Repositories, 2014, pp. 92–101, doi: 10.1145/2597073.2597074.

10. M. Golzadeh, A. Decan, D. Legay, and T. Mens, “A ground-truth dataset and classification model for detecting bots in GitHub Issue and PR comments,” J. Syst. Softw., vol. 175, p. 110,911, May 2021, doi: 10.1016/j.jss.2021.110911.

11. M.-A. Storey and A. Zagalsky, “Disrupting developer productivity one bot at a time,” in Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng., 2016, pp. 928–931, doi: 10.1145/2930290.2983989.