IEEE Software Special Issue: Bots in Software Engineering
Editors: E. Shihab, S. Wagner, M. Gerosa, M. Wessel, J. Cabot, sw5-22@computer.org

Recognizing bot activity in collaborative software development

Mehdi Golzadeh, Tom Mens and Alexandre Decan
Software Engineering Lab, University of Mons, Belgium

Eleni Constantinou
Eindhoven University of Technology, Netherlands

Natarajan Chidambaram
Software Engineering Lab, University of Mons, Belgium

***Abstract*—Machine accounts (bots) are an integral part of collaborative software development. They automate many repetitive tasks on behalf of their human owners. Focusing on popular open-source projects on GitHub, we provide evidence that bots are regularly among the most active contributors, even though GitHub does not explicitly acknowledge the presence of several of these bots. This poses a problem for techniques that analyze human contributor activity. For example, tools for recognizing and giving credit to project members for their contributions may wrongly credit bots for their automated contributions. This calls for more accurate bot identification techniques.**

Index Terms: machine account, bot identification, contributor activity, developer contribution

Motivation and context

Distributed software development is, by definition, a collaborative effort involving many different persons, teams, organizations and companies. This highly collaborative software development process has led to the creation and widespread use of distributed versioning systems such as git, social coding platforms such as GitHub and GitLab, issue tracking tools such as Bugzilla, code reviewing tools such as Gerrit, and a plethora of continuous integration and deployment (CI/CD) services.

As witnessed by initiatives such as the CHAOSS Linux Foundation Project (https://chaoss.community) and associated software development analytics tools such as GrimoireLab (https://chaoss.github.io/grimoirelab/), it is important to assess the health of software communities by considering the activity of

each contributor. Such information is also highly relevant to credit and recognise project contributors based on their activity [1], and to allow employers to identify appropriate new team members [2].

An important challenge in doing so is the presence of development robots (hereafter abbreviated to *bots*) that automate repetitive tasks to help software project contributors in their day-to-day activities. Not properly taking into account these bots may lead to incorrect or misleading conclusions, especially if such bots belong to the top contributors in software projects. This is likely to be the case, since bots are increasingly used to automate a wide range of activities, such as welcoming newcomers, reporting test coverage, updating dependencies, detecting vulnerabilities, supporting code review, submitting pull requests, verifying licensing issues, and so on [3].

In this article, we provide evidence that bots are regularly among the most active contributors in popular GitHub projects, even though GitHub does not explicitly indicate these contributors as being bots. This can be problematic for tools that aim to credit human project contributors for their activity.

Acknowledging contributions in collaborative development

Being able to accurately assess the contributions of project participants is valuable for many purposes. Software engineering researchers involved in empirical analyses of socio-technical activity and productivity in software projects need such data in order to understand and improve the development processes [4]. Prospective employers may want to analyze developer activity profiles in order to identify skilled developers that match their job openings as closely as possible [2]. Individual contributors may desire to get proper credit and visibility for their –often significant– contributions in the software projects they are involved in. They may want to use this recognition for career promotion or even to get some kind of financial support for the –often voluntary– work they spend on a project [1].

The way in which recognition is credited can differ a lot depending on the considered community. For example, OpenStack recognizes unsung heroes by discerning community contributor awards. GitHub has a similar GitHub Stars program. Initiatives such as GitHub Sponsors allow companies to sponsor open-source projects in order to help the project contributors get the recognition they deserve. Tools such as *SourceCred*[1] aim to support communities in measuring and rewarding value creation.

It is challenging to correctly determine the contributions of each project member [5]. A first challenge concerns which types of contributions should be considered [6], [7]. Typically, automated tools for identifying contributions (such as *octohattrack*[2] or *auto add contributors*[3]) provide only an impartial picture, as they tend to focus only on the types of activity that are discernible from the social coding platform (e.g., commits, pull requests, or code reviews). Other types of important contributions (e.g., finance, infrastructure, community management) are therefore often ignored [8].

A second challenge concerns how to identify contributors. If the same contributor uses multiple distinct accounts, or if the same account is shared by multiple contributors, identity merging and matching techniques are needed [9].

Another challenge concerns how to measure activity. The real effort of contributors can only be approximated. For example, the number and size of code commits could be used as a proxy of the coding effort, but does not reflect the time required to produce such a commit, since this may depend on many external factors. Moreover, such a proxy is unable to distinguish between manual or automated activity.

Last but not least, contributors may, and regularly do, use (some of) their social coding accounts to allow automated tools (i.e., bots) to carry out repetitive activity on their behalf. Whether this is intentional or not, such usage of bots that carry out tasks on behalf of their owner can disrupt the aforementioned accreditation and recognition need. Indeed, it would be unfair to give the same recognition to a contributor whose contributions are primarily due to a bot that is committing on his/her behalf, as compared to a contributor that has invested a similar effort manually. On the other hand, there is nothing wrong

[1] https://sourcecred.io
[2] https://github.com/LABHR/octohatrack
[3] https://github.com/marketplace/actions/auto-add-contributors

Published by the IEEE Computer Society

with contributors that try to increase their productivity by automating some of their repetitive tasks, as long as this is not intentionally done to artificially inflate one's activity. Whether and how to give proper recognition to project contributors remains an open and difficult question.

Distinguishing bots from humans

A first and important step to give proper recognition to project contributors consists of distinguishing human activity from bot activity. GitHub allows project contributors to discern whether certain types of activity are automated, specifically for GitHub Apps and GitHub Actions. According to the GitHub terms of service, bots are not permitted to register new GitHub accounts. However, things get more complex, since humans are permitted to set up *machine accounts* to perform automated tasks (such as a continuous integration bot), provided that a human owning the account accepts the responsibility for its actions. The problem is that the GitHub API does not allow to distinguish all such machine accounts from ordinary user accounts corresponding to real human activity. As a consequence, tools that want to benefit from distinguishing human users from machine users (i.e., bots) have a hard time doing so. For example, among the available tools to accredit and acknowledge contributors, *SourceCred* and *contributors-list*[4] are limited in separating human and bot contributors by relying on the GitHub API and on a user-defined list of machine accounts to do so.

This is where bot identification tools could come to the rescue. Such tools aim to distinguish bots from humans in GitHub accounts on the basis of their behaviour. The way of doing so can be quite diverse: it can be based on differences in the commenting patterns made by bots [10], on naming conventions, or on commit activity patterns [11]. Examples of such tools are *BoDeGHa*[5] that relies on comments made in pull requests and issues, and *BoDeGiC*[6] that relies on git commit messages.

Using bot identification tools makes it easier to dissociate bot accounts from human accounts, but can still lead to incorrect detections, notably when accounts are invoved in a mix of manual human activity and automated machine-generated activity [12]. Although there is still room for improving bot identification tools [13], they can already be very helpful in identifying bots, especially in large repositories.

Some evidence of bot contributions

In order to justify the need for properly identifying bot activity in collaborative software development projects, we provide some evidence of the presence of bots among the top contributors in popular open-source projects on GitHub. We selected 10 large and active open-source projects for popular programming languages (e.g., JavaScript, Java, Python, Rust). The list notably includes: *VueJS*[7], a very popular front-end framework for JavaScript with more than 40K dependent projects on NPM; *Servo*[8], an experimental browser engine written in Rust that has more than 1K contributors and nearly 40K commits; and *Cucumber-JVM*, a Java implementation of the popular test framework that has more than 53K dependent projects on GitHub.

We relied on the GitHub API to retrieve the contributors with the highest number of commits in these 10 projects, as well as their account type (i.e., user or bot) as reported by the GitHub API on 9 November 2021. In contrast to prior work, which has focused on the ability of machine learning classifiers to correctly identify bots, this work focuses on the possible impact of bots, that are not reported as such by the GitHub API, on the attribution to contributors.

Fig. 1 depicts the top 20 contributors to these 10 popular software projects, ranked in decreasing order of activity. Contributors that are responsible for at least 1% of all commits are highlighted. We classified the contributors into three categories: *human users*, *labeled bots* as reported by the GitHub API, and *unidentified bots* that were not reported as bots by GitHub. This classification was confirmed through a manual inspection of their activities by two authors of this paper.

The figure shows that the considered projects have between one and three bots among the top

[4]https://giters.com/wow-actions/contributors-list
[5]https://github.com/mehdigolzadeh/BoDeGHa
[6]https://github.com/mehdigolzadeh/BoDeGiC
[7]https://github.com/vuejs/vue
[8]https://github.com/servo/servo

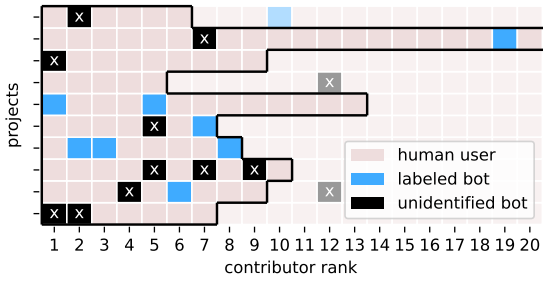

Figure 1. Bots observed in the top 20 most active committers in 10 popular open-source projects.

20 contributors. However, only less than half of the bots (9 out of 21) are reported as such by the GitHub API. The results are even more striking if we focus on the subset of contributors responsible for at least 1% of all commits: the overwhelming majority of the bots (18 out of 21) belong to those contributors and most of them (10 out of 18) are not labeled as bots by GitHub. On average, the bots are responsible for nearly one fifth of all commits in these projects.

Interestingly, we also found that some projects had explicitly credited and acknowledged bots in the list of "*people* that contributed to the project". While explicitly crediting and acknowledging contributors may encourage them to continue to contribute, the presence of bots in the contributor list may be perceived as a lack of consideration or respect towards human contributors.

What's next?

The results of our analysis revealed that bots play an undeniable role in large collaborative software development projects. These bots seem to carry out a significant amount of work, as many of them belong to the most active project contributors. Nevertheless, many bot accounts are not labeled as such by GitHub. Understanding why they are not labeled as bots remains an open question.

Having unidentified bots among the most active contributors may be problematic. For example, the presence of such bots in a contributor list may cause difficulties when changes in the project's Contributor License Agreement (CLA) are required, since such changes require the explicit approval of all human contributors. It also becomes more difficult to give due credit to human contributors for their activities, and could even lead to bots (or rather the human owners of the associated machine accounts) receiving financial compensation for their effort.

While more advanced bot detection techniques exist [10], [11], and even if they are more reliable than the GitHub API to identify bots, they are still not sufficient to accurately capture all bots [13]. Moreover, existing bot identification techniques mostly take into account specific coding-related activity types (e.g., commits, pull requests, issues, etc.). In order to cope with the diversity of contributions in collaborative development projects [5]–[7], there is a need for techniques that take into account a considerably wider range of activities (e.g., discussions, bug handling, infrastructure and community management, even financial contributions).

As a consequence, maintainers currently have no choice but to manually maintain a list of active bots in their repository, by manually inspecting contributors' activities on a regular basis. While this option is feasible for smaller repositories, it is impractical to do such a manual inspection in repositories with a large number of contributors and activities. This highlights the need to rely on automatic bot identification and in turn calls for more research on accurate bot identification techniques.

Moreover, since we expect bots to become more complex and more sophisticated in the range of development activities they support and automate, there is also a need for exploiting machine learning and artificial intelligence techniques to properly detect and acknowledge the presence of bots and their specific activity patterns.

Acknowledgements.

This research is supported by DigitalWallonia4.AI research project ARIAC (grant number 2010235), as well as by the Fonds de la Recherche Scientifique – FNRS under grants number O.0157.18F- RG43 (Excellence of Science project SECO-ASSIST) and T.0017.18.

REFERENCES

1. Il-Horn Hann, Jeff Roberts, Sandra Slaughter, and Roy Fielding. Economic incentives for participating in open

source software projects. *International Conference on Information Systems*, (33), 2002.

2. Claudia Hauff and Georgios Gousios. Matching GitHub developer profiles to job advertisements. In *Working Conference on Mining Software Repositories*, pages 362–366. IEEE/ACM, 2015.
3. Mairieli Wessel, Bruno Mendes de Souza, Igor Steinmacher, Igor S. Wiese, Ivanilton Polato, Ana Paula Chaves, and Marco A. Gerosa. The power of bots: Characterizing and understanding bots in OSS projects. *Human-Computer Interaction*, 2, 2018.
4. Zhifang Liao, Xiaofei Qi, Yan Zhang, Xiaoping Fan, and Yun Zhou. How to evaluate the productivity of software ecosystem: A case study in GitHub. *Scientific Programming*, 2020.
5. Eirini Kalliamvakou, Georgios Gousios, Diomidis Spinellis, and Pouloudi Nancy. Measuring developer contribution from software repository data. In *Mediterranean Conference on Information Systems*, 2009.
6. J. Cheng and J. L. C. Guo. Activity-based analysis of open source software contributors: Roles and dynamics. In *International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 11–18. IEEE/ACM, 2019.
7. Javier Luis Cánovas Izquierdo and Jordi Cabot. On the analysis of non-coding roles in open source development. *Empirical Software Engineering*, 27(1), 2021.
8. Jean-Gabriel Young, Amanda Casari, Katie McLaughlin, Milo Z. Trujillo, Laurent Hébert-Dufresne, and James P. Bagrow. Which contributions count? Analysis of attribution in open source. In *International Conference on Mining Software Repositories*, pages 242–253. IEEE/ACM, 2021.
9. Mathieu Goeminne and Tom Mens. A comparison of identity merge algorithms for software repositories. *Science of Computer Programming*, 78(8):971–986, 2013.
10. Mehdi Golzadeh, Alexandre Decan, Damien Legay, and Tom Mens. A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments. *Journal of Systems and Software*, 175, 2021.
11. Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. Detecting and characterizing bots that commit code. In *International Conference on Mining Software Repositories*, pages 209–219. ACM, 2020.
12. Nathan Cassee, C. Kitsanelis, Eleni Constantinou, and Alexander Serebrenik. Human, bot or both? A study on the capabilities of classification models on mixed accounts. In *International Conference on Software Maintenance*. IEEE, 2021.
13. Mehdi Golzadeh, Alexandre Decan, and Natarajan Chidambaram. On the accuracy of bot identification tools. In *International Workshop on Bots in Software Engineering*, 2022.

Author biographies

Mehdi Golzadeh obtained his master's degree in Information Technology Engineering in 2015 from the University of Tehran, Iran. He has seven years of industry experience as a developer and group leader. Since 2019, he is a PhD student at the Software Engineering Lab of the University of Mons (Belgium), in the context of the Belgian FNRS-FWO Excellence of Science research project SECO-Assist. He carries out empirical software engineering research, with a focus on the social aspects of online coding and identifying automated behaviors. Contact him at mehdi.golzadeh@umons.ac.be
Address: University of Mons, Avenue Maistriau 15, 7000 Mons, Belgium

Tom Mens is full professor and head of the Software Engineering Lab at the University of Mons, Belgium. His main research interests are empirical analysis of, and tooling for, open-source software ecosystems. He published numerous scientific articles on this topics in peer-reviewed international journals, conferences and workshops. He co-edited two Springer books on software evolution. He was program chair of ICSM 2013, CSMR 2012 and CSMR 2011, keynote speaker for ICSME 2016 and invited lecturer at several international summer schools. He is project coordinator of the Belgian inter-university Excellence of Science research project SECO-Assist. He is IEEE Senior Member. Contact him at tom.mens@umons.ac.be
Address: University of Mons, Avenue Maistriau 15, 7000 Mons, Belgium

Alexandre Decan obtained a PhD in Sciences in 2013 at the Faculty of Sciences of the University of Mons (Belgium) on the subject of data quality in relational databases. He is post-doctoral researcher at the Software Engineering Lab of the University of Mons, where he has authored many well-cited publications related to the maintenance and evolution of software ecosystems. He has been actively involved in several research projects such as the UMONS Action de Recherche Concertée ECOS, the Walloon ERDF project portfolio IDEES, the FNRS-FRQ collaborative research project Seco-Health and the Belgian FNRS-FWO Excellence of Science project SECO-Assist. Contact him at alexandre.decan@umons.ac.be
Address: University of Mons, Avenue Maistriau 15, 7000 Mons, Belgium

Eleni Constantinou is an assistant professor at the Eindhoven University of Technology, Netherlands. Her main research interests include mining software repositories, software ecosystems and software evolution. She has published her results in numerous peer-reviewed journals, conferences and workshops. She has served in the program and organizing committee of several premier conferences and workshops. Contact her at e.constantinou@tue.nl
Address: Eindhoven University of Technology, De Groene Loper 5, 5612AZ Eindhoven, Netherlands

Natarajan Chidambaram obtained his master's degree in Data Science in Engineering in 2020 from the Eindhoven Uni-

Published by the IEEE Computer Society

versity of Technology, Netherlands. He has two years of industry experience as an AUTOSAR driver code developer. Since 2021, he is a PhD student at the Software Engineering Lab of the University of Mons (Belgium), in the context of research project ARIAC by DigitalWallonia4.AI. His research focuses on socio-technical analysis in collaborative open-source software development. Contact him at natarajan.chidambaram@umons.ac.be

Address: University of Mons, Avenue Maistriau 15, 7000 Mons, Belgium