Fine-Grained Developer Reification

Stefano Campanella
REVEAL @ Software Institute – USI, Lugano, Switzerland
Supervisor: Dr. Prof. Michele Lanza
Lugano, Switzerland
stefano.campanella@usi.ch

Abstract

To capture the evolution of software projects and simplify their understanding, researchers rely on metadata stored in collaborative tools such as issue trackers and version control systems. While these systems provide comprehensive data about codebases, project management practices, and contributors, developers are often abstracted in a simplistic manner, using only basic identifiers like unique usernames or commit-associated emails. This approach may overlook crucial aspects of developers' activities, potentially leading to findings based on incomplete or inconsistent data.

To address the lack of concrete characterization, we propose implementing a developer-centric model that comprehensively represents each developer and their contributions to the software development community. We intend to leverage the concept of *gamification* and present developers as avatars, akin to collectors of all the software engineering knowledge they have collected over the years. This approach will provide a more interactive and engaging way to comprehend such complex information. To do so, we analyze the activities performed by each individual across multiple projects and platforms at various levels of granularity to capture their essence and uniquely represent them accurately.

In an era where robots and artificial intelligence are growing more integrated into various aspects of the software engineering field, we want to shift the focus back to what is the most crucial link in the software development chain: **The Human**

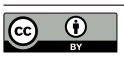
CCS Concepts

• **Software and its engineering** → **Programming teams**; • **Human-centered computing** → **Visualization**.

Keywords

Developers, Reification, Activity, Mining Software Repositories

ACM Reference Format:
Stefano Campanella. 2025. Fine-Grained Developer Reification. In *33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25), June 23–28, 2025, Trondheim, Norway*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3696630.3731463



FSE Companion '25, Trondheim, Norway

ACM ISBN 979-8-4007-1276-0/2025/06
https://doi.org/10.1145/3696630.3731463

1 Introduction

Nowadays, Open Source Software (OSS) plays an increasingly important role in driving innovation [31]. Software systems, especially in this context, are the result of collective efforts by numerous individuals with diverse backgrounds and expertise. Understanding the spread of knowledge and dynamics among developers [24] and different communities has become a crucial aspect of software development [43, 44]. The success of OSS projects in particular, strongly depends on the developers who contribute to such initiatives. For this reason, they have to be considered the most important resource available and have to be treated as such. Sharing expertise and coding skills between developers [9] is, in fact, the easiest route for OSS projects to achieve fast code improvements and efficient debugging. The activity performed by developers contributing to OSS, whether directly in the code or in lateral contexts such as issues and pull requests, represents the heart of the software itself and can ultimately provide a summary of a developer's knowledge [47].

When studying software systems, OSS is often used as the main source of information. It usually provides data about the codebase of the software, entities related to project management practices, and information about the contributors of the project. Despite the importance of the developers, during the data collection process, researchers often ignore developer-related information and focus on the one obtained from code and other natural language-based elements (*e.g.,* issues and pull requests). When mining software repositories and collaborative development platforms (*e.g.,* GitHub), they abstract the concept of the developer in the simplest way possible, such as by using the cloud-based platform's unique username or the email used to make a commit. Because of this behavior, developers-oriented studies that try to measure phenomena such as developers' productivity and expertise risk losing valuable information regarding the complete activity of the developers, ultimately producing results based on incomplete or inconsistent data [6, 23].

Our goal is to provide comprehensive details about developers by examining their work patterns across multiple projects and different platforms over time. We aim to achieve this through a reification process, which involves transforming abstract concepts into concrete representations within a system. This process involves converting high-level ideas into explicit data models or objects in a programming language, making previously implicit structures explicit and manipulable. As a result of this process, our work will enable a more flexible and concrete interaction with developers and their activity by explicitly formulating and making available their figures for conceptual (logical or computational) manipulation. Building on our findings, we will be able to shape the figure of the developer in more comprehensible and achievable ways, such as by developing a gamification system that revolves around, but not solely encompasses, the developers.

2 Background and Related Work

Researchers measured different actions and characteristics of developers over the years. The data used to perform such measurements is primarily collected from version control systems [28]. Given the heterogeneous nature of the software development process, such information is often integrated with additional data from collaborative environments [23], technical forums [46], and development tools (*e.g.,* IDEs) [27, 40]. From this data, they were able to measure characteristics such as activity [36, 45], code quality [12], and experience [7] and proposed numerous amount of metrics. On the downside, these metrics provide very different results according to slight variations [1, 33, 34].

While there are traces of large-scale systems that try to aggregate data across multiple sources [29], because of the long-lasting challenges of handling and cross-referencing diverse data sources [21], most of the works in this context focus on single elements such as commits [3, 10] or lines of code [17]. Leveraging such information, researchers have tried to define ways of identifying core and peripheral roles among contributors in software development communities [22]. Others have explored creating professional CVs directly from source code repositories [5, 20]. Some studies focus on defining roles based on contributors' activities and responsibilities [11, 30], while others present automated methods to extracting developers' expertise [14, 15]. Additionally, studies have identified specific skills and competencies crucial for effective software development [25]. Recommendation systems have also been developed to provide tailored suggestions for tasks and roles based on developer profiles and project needs [9, 26, 47]. Finally, visualization techniques can also offer insightful representations of development activities [16].

Gamification, a captivating follow-up to the reification process, has gained significant traction in recent years, despite the challenges involved. Researchers have made notable contributions in three distinct areas: (i) providing design frameworks [39], (ii) developing novel methods for evaluating the outcomes of gamification processes [42], and (iii) applying gamification to various domains of software engineering [13, 41]. The primary application of gamification in software engineering lies in the educational context [2], where it has demonstrated its effectiveness as a supportive tool to enhance the learning process [32].

3 Research Focus

The reification of the developer's figure is the core of our research. This process will elevate developers to first-class citizens in the context of software engineering analysis and observation. Achieving this goal requires a stepwise refinement, where we progressively dissect and model the developer's actions, decisions, and contributions across the software development lifecycle. This work will require a twofold approach: (a) investigating data available in git, which can give us more insight into the activity performed by the individuals (*i.e.,* going deeper into the code), and (b) integrating activity performed outside of the versioning system, such as interactions with collaborative tools elements (*e.g.,* issues). Finally, based on the model obtained thanks to this process, it will be possible to create new ways of representing developers' characteristics, such as a gamification system.

At first, we will analyze high-level information, such as commit frequency and volume of contributions, to characterize the level of activity and involvement of the developer in the context of OSS, and in general with public code. Then, by going deeper into details, we will look at programming languages, frameworks, and technologies employed, to infer the roles the developer covers over his projects (*e.g.,* front-end development). Once we obtain a coarse-grained characterization of the developer, we can delve into the semantic meaning of the code he pushes to remote repositories. Analyzing code opens up different routes: (i) we can focus on the meaning of the commit, by detecting and characterizing different common coding tasks such as refactoring, and (ii) we can also analyze the quality of the code committed, and characterize the overall level of quality for the contributions made by a developer. Additionally, by analyzing information that is not code-related by definition, such as contributions to documentation and DevOps practices, we can introduce additional information on the characteristics of a developer, which were not measurable otherwise. Finally, by focusing on community interactions, we can also get information on the position of the developer in the community, thus broadening the model with an additional level of information.

At the end of this process, we will be able to mold and control most of the information about a developer. This will also allow us to give control of this information to the developer itself. For instance, we could incorporate automatically generated information into the model, thereby augmenting its overall knowledge. Alternatively, we could eliminate information that the developer does not consider as their own. By implementing these measures, developers will gain complete control over their information, which is stored in software engineering artifacts.

As previously anticipated, once the reification process is mature enough, we envision the possibility of building a gamification system by mapping the newly obtained developer's profile to an "Avatar". Drawing inspiration from character creation and evolution in role-playing games, such elements can be mapped to different objects such as classes (roles), equipment (used languages), and skills (expertise). Similarly, common software engineering concepts and artifacts, such as technical debt and repositories, can be translated as levels or monsters that require special characteristics and skill sets to be overtaken or defeated. Finally, elements such as issues can be translated to missions that the developers have to accomplish by contributing to a repository to collect experience and level up their avatars. By introducing a reward system as a result of overcoming obstacles or reaching certain goals, gamification will enable a more rewarding experience when contributing to open source (OS) projects and also represent their achievements engagingly. For instance, by rewarding active participation in community discussions, the system will encourage developers to keep participating in this kind of activity while reaching progressive milestones, such as increasingly higher amounts of commits, which will encourage developers to keep actively working on OS projects.

By reifying developers' activities, we aim to transform initial insights into a tangible, interpretable entity representing their contributions, roles, and commitments. Furthermore, we seek to provide an interactive way to engage with these entities. This approach will empower developers with full control over their information, stored in version control systems and collaborative platforms.

3.1 Research Agenda

Following our ideas, we want to pursue the reification of developers by following several steps, with the twofold aim of being stand-alone contributions that enhance the ability to characterize developers and also a brick into the overall final gamified system. To evaluate the reification of the developers, we plan to perform multiple iterations of a small-scale study on a group of projects of which we are aware of the team's structure, roles, and responsibilities; followed by a large-scale study where we will ask developers in the OS community to validate our characterization made by using publicly available contributions. Once we obtain the full characterization, we will start the gamification process that will consist of mapping such characteristics into avatars, as previously described. Together with such avatars, we will also introduce entities related to elements such as repositories and issues. This system will then be evaluated using approaches to evaluate gamified systems already available in the literature (*e.g.,* GATUGU [42]).

The progression of the reification will be structured as follows:

- **Coarse-grained Characterization:** By analyzing high-level information such as programming languages, frameworks, and tools, we will characterize the roles and contributions of the developers. By focusing on Interaction in Issues and Pull Requests, we will also introduce community-related roles.
- **Fine-grained Characterization:** Focusing on the kind of contribution (*e.g.,* code, documentation, testing) and the content of the contributions (*e.g.,* edited code), we will infer and characterize areas of expertise and specializations. Additionally, we will enrich the model with collateral code information (*e.g.,* quality, self-declared technical debt, test coverage).
- **Evolution of the Developers:** Once the model is comprehensive on most of the characteristics of a developer, we will leverage it to study and understand the evolution of the roles and abilities of the developer during their professional journey.
- **Gamification system:** Leveraging the model obtained from the previous steps, we will build a gamification system that represents the developers and other elements of the software engineering domain to achieve a more engaging way of collecting, representing, and creating new activities.

Problems and Common Challenges. The increasing popularity of LLM-based code assistants will require ad-hoc techniques able to distinguish between code generated by the assistant and code implemented by the developer to accurately characterize the developer's behavior [35, 49]. Furthermore, when dealing with individuals and their identities, privacy is paramount [18], especially if we consider that gamified systems can be misused to collect a vast amount of information about the players [39]. On the contrary, if properly implemented, a gamification system that gives rewards and full control of the information to the owners of the data can mitigate privacy concerns. Finally, it is of fundamental importance that the final product is designed in a way that the system does not push users to engage in negative behaviors to collect more prizes or achieve the goal faster, as bad-designed measurement systems can supposedly lead developers to game the system rather than improve the actual quality of their codebases [45]. To mitigate such problems, our work will leverage available tools or eventually propose new approaches.

4 Current Research Progress

We begin the reification process by exploring developers' contributions using and enhancing existing techniques. In our initial work, we focus on the contributions of developers to single software repositories [8]. Additionally, we investigated some of the most common problems that researchers have to tackle when working on identities, such as the adoption of multiple identities by a single individual (*i.e.,* aliasing) and the detection of bot activity. Distinguishing between human [37] and automated [38, 48] activity has proven to be challenging but fundamental for the quality of the data collection and characterization of developers. Researchers have proposed various techniques, combining machine learning and simple heuristics, to merge identities. Our work leverages the combination of two state-of-the-art approaches (*i.e.,* Gambit [19], ALFAA [4]) to match identities based on a set of heuristics. As for bot detection, we present a novel approach that combines tools available in the literature. This data-cleaning process outputs a refined list of contributors, each associated with a set of commits that may have been authored by seemingly different identities.

The coding activity of these contributors is illustrated as a timeline to support the comprehension of the distribution of activity in time. Focusing on individuals revealed that the use of multiple aliases is common among developers, suggesting that it is important to tackle this problem when working with developers' identities. This study led us to a series of observations. First, we are now capable of easily observing different dynamics of the developers' activities in software projects. Second, we noticed how it is not uncommon for developers to work in groups over different stages of the repository lifetime. Third, we noticed how the majority of contributors have a relatively short duration of activity within a single project. Finally, we noticed a growing trend toward the popularity of automatically generated contributions, that span from automated fixes of dependencies to entire code changes.

Based on these first insights and the techniques we were able to leverage, we are now in the process of further investigating such activity from two distinct points of view: (i) analyzing the effect of such activity on the overall life of a software repository and (ii) analyzing developers' code activity in greater detail by focusing on single commit-based information over different projects. In addition, we are laying the foundations of the gamification system.

5 Conclusion

In the dynamic world of software development, individual developers are often overlooked as they get lost in a sea of code and collaboration. However, it is their unique skills, experiences, and contributions that ultimately drive project success. In a world where it is increasingly difficult to distinguish between human and automated activity, focusing on developers will place more emphasis on what a developer can achieve. By reifying developers as individuals with distinct characteristics, we aim to lay the foundations for developer-oriented research where the figure of the human in the context of software engineering gets the importance it deserves. Finally, we will obtain complete control over the figure of the developer and create new ways of interacting with such entities and a systematic definition of clear characteristics that can define the ability and experience of a developer.

References

[1] Waseem Ahmed and Ahmed Harbaoui. 2024. Is This Code the Best? Or Can It Be Further Improved? Developer Stats to the Rescue. *IEEE Access* 12 (2024), 144395–144411.

[2] Iñigo Aldalur. 2024. Gamifying software engineering subject to enhance the quality of knowledge. *Software: Practice and Experience* 54, 12 (2024), 2299–2315.

[3] Juan Jose Amor, Gregorio Robles, and Jesus M. Gonzalez-Barahona. 2006. Effort Estimation by Characterizing Developer Activity. In *International Workshop on Economics Driven Software Engineering Research (EDSER)*. ACM, 3–6.

[4] Sadika Amreen, Audris Mockus, Russell Zaretzki, Christopher Bogart, and Yuxia Zhang. 2020. ALFAA: Active Learning Fingerprint based Anti-Aliasing for correcting developer identity errors in version control systems. *Empirical Software Engineering* 25, 2 (2020), 1136–1167.

[5] Daniel Atzberger, Nico Scordialo, Tim Cech, Willy Scheibel, Matthias Trapp, and Jürgen Döllner. 2022. CodeCV: Mining Expertise of GitHub Users from Coding Activities. In *International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 143–147.

[6] Christian Bird, Peter C Rigby, Earl T Barr, David J Hamilton, Daniel M German, and Prem Devanbu. 2009. The Promises and Perils of Mining Git. In *Working Conference on Mining Software Repositories (MSR)*. IEEE, 1–10.

[7] Renata Brasil-Silva and Fábio Levy Siqueira. 2022. Metrics to quantify software developer experience: a systematic mapping. In *Symposium on Applied Computing (SAC)*. ACM, 1562–1569.

[8] Stefano Campanella and Michele Lanza. 2024. Hidden in the Code: Visualizing True Developer Identities. In *Working Conference on Software Visualization (VISSOFT)*. IEEE, 24–35.

[9] Gerardo Canfora, Massimiliano Di Penta, Rocco Oliveto, and Sebastiano Panichella. 2012. Who is going to mentor newcomers in open source projects?. In *Symposium on the Foundations of Software Engineering (FSE)*. ACM, 44.

[10] Andrea Capiluppi and Daniel Izquierdo-Cortazar. 2013. Effort Estimation of FLOSS Projects: A Study of the Linux Kernel. *Empirical Software Engineering* 18, 1 (2013), 60–88.

[11] Eleni Constantinou and Georgia M. Kapitsaki. 2016. Developers Expertise and Roles on Software Technologies. In *Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 365–368.

[12] Patricia Rucker de Bassi, Gregory Moro Puppi, Pedro Henrique Banali, and Emerson Cabrera Paraiso. 2018. Measuring Developers' Contribution in Source Code using Quality Metrics. In *International Conference on Computer Supported Cooperative (CSCWD)*. IEEE, 39–44.

[13] Gabriela Martins de Jesus, Fabiano Cutigi Ferrari, Daniel de Paula Porto, and Sandra Camargo Pinto Ferraz Fabbri. 2018. Gamification in Software Testing: A Characterization Study. In *Brazilian Symposium on Systematic and Automated Software Testing (SAST)*. ACM, 39–48.

[14] Tapajit Dey, Andrey Karnauch, and Audris Mockus. 2021. Representation of Developer Expertise in Open Source Software. In *International Conference on Software Engineering (ICSE)*. IEEE, 995–1007.

[15] Anett Fekete, Máté Cserép, and Zoltán Porkoláb. 2021. Measuring Developers' Expertise Based on Version Control Data. In *International Convention on Information, Communication and Electronic Technology (MIPRO)*. IEEE, 1607–1612.

[16] Anett Fekete and Zoltán Porkoláb. 2024. Using Version Control Information to Visualize Developers' Knowledge. *Acta Cybernetica* 26, 3 (2024), 431–454.

[17] Matheus Silva Ferreira, Luana Almeida Martins, Paulo Afonso Parreira Júnior, and Heitor A. X. Costa. 2020. How is the Work of Developers Measured? An Industrial and Academic Exploratory View. *Journal of Software Engineering Research and Development* 8 (2020), 1–20.

[18] Nicolas E. Gold and Jens Krinke. 2022. Ethics in the mining of software repositories. *Empirical Software Engineering (EMSE)* 27, 1 (2022), 17.

[19] Christoph Gote and Christian Zingg. 2021. Gambit - An Open Source Name Disambiguation Tool for Version Control Systems. In *International Conference on Mining Software Repositories (MSR)*. IEEE, 80–84.

[20] Gillian J. Greene and Bernd Fischer. 2016. CVExplorer: identifying candidate developers by mining and exploring their open source contributions. In *International Conference on Automated Software Engineering (ASE)*. ACM, 804–809.

[21] Ahmed E. Hassan. 2008. The road ahead for Mining Software Repositories. In *2008 Frontiers of Software Maintenance*. IEEE, 48–57.

[22] Mitchell Joblin, Sven Apel, Claus Hunsen, and Wolfgang Mauerer. 2017. Classifying developers into core and peripheral: an empirical study on count and network metrics. In *International Conference on Software Engineering (ICSE)*. IEEE / ACM, 164–174.

[23] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. Germán, and Daniela E. Damian. 2014. The promises and perils of mining GitHub. In *Working Conference on Mining Software Repositories (MSR)*. ACM, 92–101.

[24] Stefan Koch and Georg Schneider. 2002. Effort, Co-operation and Co-ordination in an Open Source Software Project: GNOME. *Information Systems* 12, 1 (2002).

[25] Stratos Kourtzanidis, Alexander Chatzigeorgiou, and Apostolos Ampatzoglou. 2020. RepoSkillMiner: Identifying software expertise from GitHub repositories using Natural Language Processing. In *International Conference on Automated Software Engineering (ASE)*. IEEE, 1353–1357.

[26] Sandeep Kaur Kuttal, Xiaofan Chen, Zhendong Wang, Sogol Balali, and Anita Sarma. 2021. Visual Resume: Exploring developers' online contributions for hiring. *Information and Software Technology* 138 (2021), 106633.

[27] Zhixing Li, Yue Yu, Tao Wang, Gang Yin, ShanShan Li, and Huaimin Wang. 2021. Are You Still Working on This? An Empirical Study on Pull Request Abandonment. *Transactions on Software Engineering (TSE)* 48, 6 (2021).

[28] Jalerson Lima, Christoph Treude, Fernando Marques Figueira Filho, and Uirá Kulesza. 2015. Assessing Developer Contribution with Repository Mining-based Metrics. In *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 536–540.

[29] Yuxing Ma, Chris Bogart, Sadika Amreen, Russell Zaretzki, and Audris Mockus. 2019. World of code: an infrastructure for mining the universe of open source VCS data. In *International Conference on Mining Software Repositories (MSR)*. IEEE / ACM, 143–154.

[30] João Eduardo Montandon, Marco Túlio Valente, and Luciana Lourdes Silva. 2021. Mining the Technical Roles of GitHub Users. *Information and Software Technology* 131 (2021), 106485.

[31] Hussan Munir, Johan Linåker, Krzysztof Wnuk, Per Runeson, and Björn Regnell. 2018. Open Innovation Using Open Source Tools: A Case Study at Sony Mobile. *Empirical Software Engineering* 23, 1 (2018), 186–223.

[32] Vincenzo Di Nardo, Riccardo Fino, Marco Fiore, Giovanni Mignogna, Marina Mongiello, and Gaetano Simeone. 2024. Usage of Gamification Techniques in Software Engineering Education and Training: A Systematic Review. *Computers* 13, 8 (2024), 196.

[33] Vincenzo Orrei, Marco Raglianti, Csaba Nagy, and Michele Lanza. 2023. Contribution-Based Firing of Developers?. In *Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM, 2062–2066.

[34] Norman Peitek, Sven Apel, Chris Parnin, André Brechmann, and Janet Siegmund. 2021. Program Comprehension and Code Complexity Metrics: An fMRI Study. In *International Conference on Software Engineering (ICSE)*. IEEE, 524–536.

[35] Musfiqur Rahman, SayedHassan Khatoonabadi, Ahmad Abdellatif, and Emad Shihab. 2024. Automatic Detection of LLM-generated Code: A Case Study of Claude 3 Haiku. *ArXiv* abs/2409.01382 (2024), 20 pages.

[36] Gregorio Robles, Andrea Capiluppi, Jesús M. González-Barahona, Björn Lundell, and Jonas Gamalielsson. 2022. Development Effort Estimation in Free/Open Source Software from Activity in Version Control Systems. *Empirical Software Engineering* 27, 6 (2022), 135.

[37] Gregorio Robles and Jesús M. González-Barahona. 2005. Developer Identification Methods for Integrated Data from Various Sources. In *International Workshop on Mining Software (MSR)*. ACM, 1–5.

[38] Sivasurya Santhanam, Tobias Hecking, Andreas Schreiber, and Stefan Wagner. 2022. Bots in Software Engineering: A Systematic Mapping Study. *PeerJ Computer Science* 8 (2022), e866.

[39] Tommaso Dal Sasso, Andrea Mocci, Michele Lanza, and Ebrisa Mastrodicasa. 2017. How to gamify software engineering. In *International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 261–271.

[40] Martin Schroer and Rainer Koschke. 2024. Understanding and Evaluating Developer Behaviour in Programming Tasks. In *Workshop on Integrated Development Environments (IDE)*. ACM, 35–39.

[41] Klaas-Jan Stol, Mario Schaarschmidt, and Shelly Goldblit. 2022. Gamification in software engineering: the mediating role of developer engagement and job satisfaction. *Empirical Software Engineering* 27, 2 (2022), 35.

[42] Jakub Swacha, Ricardo Queirós, and José Carlos Paiva. 2023. GATUGU: Six Perspectives of Evaluation of Gamified Systems. *Information* 14, 2 (2023), 136.

[43] Damian A. Tamburri, Patricia Lago, and Hans van Vliet. 2013. Uncovering Latent Social Communities in Software Development. *IEEE Software* 30, 1 (2013), 29–36.

[44] Damian A. Tamburri, Fabio Palomba, and Rick Kazman. 2021. Exploring Community Smells in Open-Source: An Automated Approach. *Transactions on Software Engineering* 47, 3 (2021), 630–652.

[45] Christoph Treude, Fernando Marques Figueira Filho, and Uirá Kulesza. 2015. Summarizing and measuring development activity. In *Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*. ACM, 625–636.

[46] Sri Lakshmi Vadlamani and Olga Baysal. 2020. Studying Software Developer Expertise and Contributions in Stack Overflow and GitHub. In *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 312–323.

[47] Yao Wan, Liang Chen, Guandong Xu, Zhou Zhao, Jie Tang, and Jian Wu. 2018. SCSMiner: mining social coding sites for software developer recommendation with relevance propagation. *World Wide Web* 21, 6 (2018), 1523–1543.

[48] Mairieli Santos Wessel, Marco Aurélio Gerosa, and Emad Shihab. 2022. Software Bots in Software Engineering: Benefits and Challenges. In *International Conference on Mining Software Repositories (MSR)*. ACM, 724–725.

[49] Tong Ye, Yangkai Du, Tengfei Ma, Lingfei Wu, Xuhong Zhang, Shouling Ji, and Wenhai Wang. 2024. Uncovering LLM-Generated Code: A Zero-Shot Synthetic Code Detector via Code Rewriting. *ArXiv* abs/2405.16133 (2024), 12 pages.