Reviewing Career Paths of the OpenStack Developers

Perry van Wesel*, Bin Lin†, Gregorio Robles‡, Alexander Serebrenik*
*Eindhoven University of Technology, The Netherlands, p.m.m.v.wesel@student.tue.nl, a.serebrenik@tue.nl
†Università della Svizzera italiana (USI), Switzerland, bin.lin@usi.ch
‡Universidad Rey Juan Carlos, Spain, grex@gsyc.urjc.es

***Abstract*—Career perspectives are known to motivate software engineers. However, so far, career perspectives have been mostly studied within traditional software development companies. In our work we take a complementary approach and study career paths of open source developers, focusing on their advancement through the code review hierarchy, from developers to reviewers and further to core reviewers.**

To gain understanding of code review career paths we conduct an exploratory case study of the OpenStack community. Based on the case study we have publicized anonymized research data and formulated four hypotheses pertaining to career paths of contributors in modern multi-company open source projects. We conjecture that (i) developers and reviewers are separate subpopulations with little movement between them, (ii-a) the turnover of the core reviewers is high and rapid, (ii-b) companies are interested in having core reviewers among their staff, and (iii) being a core reviewer is beneficial for career.

Validity of those hypotheses in other multi-company open source projects should be investigated in the follow-up studies.

I. Introduction

Career path-related prospects have been repeatedly reported as one of the motivators of software engineers: in the systematic literature review conducted by Beecham et al. [1] and covering the period from 1980 to 2006, the authors have identified 15 papers relating career perspectives to software engineers' motivation, while a more recent literature review of França et al. [2] has further identified nine additional papers.

While most of these studies refer to developers working on traditional *closed-source* software or to the benefits that Open Source contributors might have in terms of career advancement in traditional organizations [3], few have considered advancement within Open Source communities [4].

Previous research on different roles found in Open Source Software projects is known [5], [6]. However, since many Open Source projects do not have formally defined roles, "roles" are more commonly interpreted as contributors being more or less active [7], more or less experienced ones [8], [9], or choosing to specialize on different kinds of activities, e.g., coding, localization [7], bug-reporting/testing [10].

With the professionalization of the Open Source developer force, many developers have started to pursue a professional career as an Open Source developer. This is especially the case in those software ecosystems that are the joint effort of many industrial partners, such as the ones in OW2,[1] or OpenStack,[2]. Even if roles and paths have been identified in Open Source projects previously [5], the current scenario and career perspectives of developers have changed significantly. While ten years ago the only *professionalized* tasks corresponded to activities that *volunteer* developers did not want to perform and/or require major dedication, i.e., release managers or maintainers of strategic components, nowadays many projects count on a *professional* workforce. In such a scenario, the career possibilities and paths that an ecosystem offers may serve as a factor of attraction and retention of highly specialized developers.

To get insights in the career paths in the Open Source communities we conduct a case study on the OpenStack project. Similarly to Kula et al. [11] we operationalize career stages through the lens of a modern code review [12], [13], and distinguish between non-reviewers (developers), reviewers and core reviewers. We observe that merely 3% of the developers become code reviewers, and 37% of those become core reviewers. Moreover, becoming a core reviewer does not take long. However, many core reviewers leave after a short period. We also observe that for both full-time and non-full-time contributors, the number of reviews increases once they become core reviewers. No such uniformity can be observed with respect to the commits authored by contributors.

Based on the case study, we have formulated four hypotheses pertaining to career paths of contributors in modern industrial Open Source projects. We conjecture that (i) developers and reviewers are separate subpopulations with little movement between them, (iii) the turnover of the core reviewers is high and relatively rapid, (ii) companies are interested in having core reviewers among their staff, and (iv) being a core reviewer is beneficial for career. These hypotheses should be confirmed or refuted by follow-up studies.

We make the anonymized data used in this study public[3].

II. Why code reviews?

As indicated in Section I, similarly to Kula et al. [11] we operationalize career stages through the lens of a modern code review [12], [13]. Indeed, anecdotal evidence suggests that companies such as GitLab consider reviewing code as one of

[1]https://projects.ow2.org/bin/view/wiki/

[2]https://www.openstack.org/

[3]https://openstack-career.github.io/data.zip

the key areas of developer expertise and individuals not having it are only hired as junior developers[4].

To further assess the validity of code reviewing as an indication of the career phase, we compare the salaries associated with code reviewing as opposed to software development in general (cf. the representation condition [14]). To this end, on July 28, 2017 we search for "software developer" as opposed to "code review" jobs on Naukri.com[5], the website positioning itself as "India's No. 1 Job Site", and determine the expected salary. We observe that "code review" jobs can be expected to pay ca. 1.4 times more than "software developer" jobs.

Rather than code reviews, one might consider commit rights as an indication of a career stage. Indeed, GitHub contributors can submit changes either via pull requests subject to review and approval, or by directly committing to the repository. The latter right is granted to the trusted developers only. However, recent studies show that even some developers having commit rights prefer to submit pull requests as means of ensuring quality of the submitted change through the associated review and approval mechanism [15]. This preference undermines the interpretation of the direct commits or pull request submissions as an indication of a career advancement or lack thereof.

As opposed to the commit rights there is no reason for the core reviewers not to use their review rights. To validate this assumption we consult the record activity overview of Russel Bryant,[6] member of the Board of Directors at OpenStack Foundation. On June 21, 2017 this overview listed 2,458 contributors that have performed a review at some point, and 1,765 of them have performed at least one review in the last 180 days. Russel Bryant has explicitly marked 329 contributors as core reviewers; among them 317 (96%) exercise the associated right of approving/rejecting changes. Therefore, we believe that the reviewing rights are a more trustworthy indication of a career phase than the commit rights.

III. Research method

To gain understanding of the career paths of software developers we focus on the code reviewing activities and conduct an exploratory case study [16] of the OpenStack contributors. OpenStack can be seen as a "paradigmatic" case [17] of a modern multi-company Open Source Software project [18], [19], involving over 500 companies. The data are extracted from the MySQL dumps acquired from the Bitergia OpenStack dashboard[7] at September 1, 2016.

1) Identifying contributors: The MySQL dump contains two databases recording information about the contributors committing and reviewing activities, respectively. Activities are recorded per account, i.e., activities of a contributor using different mail addresses are kept separate. Moreover, accounts in the commit activity database are different from the accounts in the reviewing activity database. To get insights in the activities of the individual contributors, Bitergia has manually linked together accounts suspected to belong to the same person; members of the OpenStack Foundation verified correctness of the linkage. However, as this linkage process is manual, it is inherently incomplete. Therefore, we further link accounts if the associated mail addresses are identical. We opt for this conservative solution rather than more advanced techniques proposed in the literature [20], [21], [22] since the lion's share of the linkage has been already carried out manually and verified by the Open Stack foundation.

Release	Date	Release	Date	Release	Date
Austin	2010-10-21	Folsom	2012-09-27	Juno	2014-10-16
Cactus	2011-04-15	Grizzly	2013-04-04	Kilo	2015-04-30
Diablo	2011-09-22	Havana	2013-10-17	Liberty	2015-10-15
Essex	2012-04-05	Icehouse	2014-04-17	Mitaka	2016-04-07

TABLE I: OpenStack release cycle.

2) Identifying career stages: During the code review, a reviewer can cast a vote in favor or against the change proposed. Five kinds of votes are supported by Gerrit[8]: -2 (the change should be rejected), -1 (by preference the change should be rejected unless there is another reviewer that would like to accept it), 0 (no opinion), +1 (the change looks right) and +2 (the change should be integrated). Since any -2 blocks the change submission and any +2 enables it, only a restricted group of contributors are allowed to cast -2 and +2 votes. Hence, we consider three career stages of the OpenStack contributors: i) *developers* that commit but do not review, ii) *reviewers* that can cast the votes between -1 and 1, and iii) *core reviewers* that also can cast -2 and +2. Section II shows that contributors who can cast a +2/-2 vote, indeed do so.

3) Identifying career paths: To identify the career paths we identify the career stages of contributors at different periods of time. We define the "periods of time" based on the release cycle of OpenStack, i.e., two releases per year approximately six months apart. Adherence to the release process ensures that our findings are not disturbed by releases [23]. Table I summarizes OpenStack releases used to create a partitioning into periods. Release *Bexar (2011-02-03)* has been excluded due to it not adhering to the six-month cycle, causing two periods of three months between *Austin* and *Bexar* and between *Bexar* and *Cactus*. We exclude from consideration those contributors joining after release *Kilo (2015-10-15)* since the period between Kilo and the collection of the data is too short to be considered indicative of a career path.

4) Identifying full-time contributors: Following Robles et al. [24] to identify full-time contributors, we require the contributors to perform at least 9 commits between two subsequent releases. While more elaborate techniques, e.g., based on the working hours, have been proposed in the literature [25], Robles et al. [24] have introduced and validated their approach on the OpenStack data, and therefore we adhere to it.

5) Statistical analysis: To compare activity of the contributors before and after they become core reviewers we compare distributions of the number of commits and the number of

[4]https://about.gitlab.com/jobs/developer/

[5]https://www.naukri.com/

[6]http://russellbryant.net/openstack-stats/all-reviewers-180.txt

[7]http://activity.openstack.org/dash/browser/data_sources.html

[8]https://gerrit-review.googlesource.com/Documentation/config-labels.html

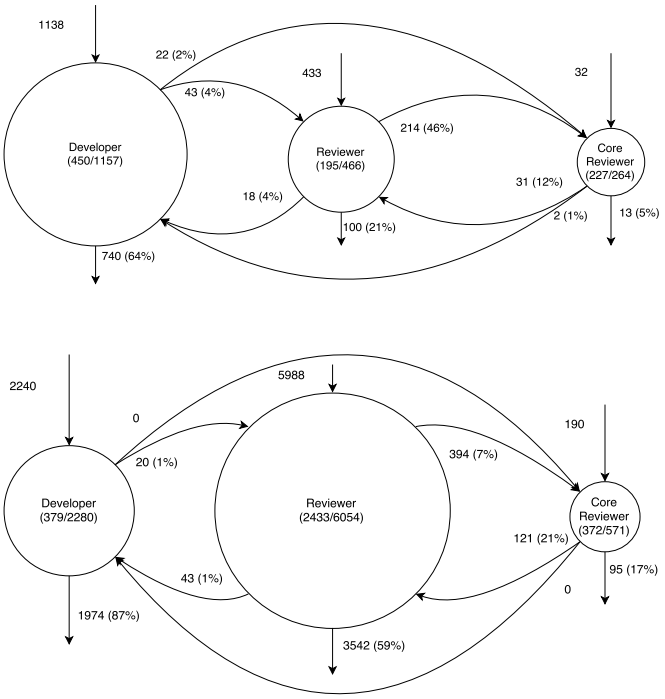


Fig. 1: Full-time (upper) and non-full-time (lower) contributors. Contributors are only considered once (even if performing the same transition several times).

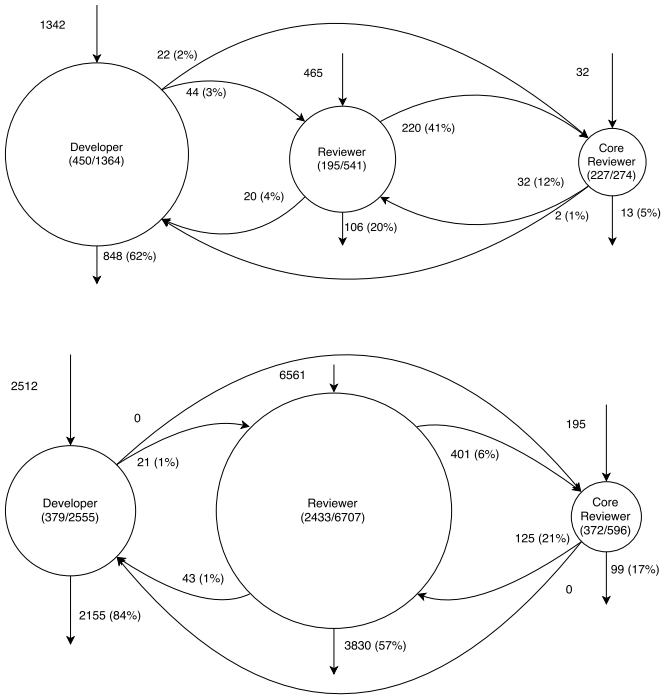


Fig. 2: Career paths of full-time (upper) and non-full-time (lower) contributors. Contributors may be considered several times (one per transition).

reviews contributed per period. We consider six different periods: two periods prior to the contributor becoming a core reviewer, the transition period an three periods after it.

To compare multiple distributions we opt for the $\widetilde{\mathbf{T}}$-procedure [26]. $\widetilde{\mathbf{T}}$ is robust against unequal population variances, respects transitivity, does not suffer form well-known problems of two-steps approaches [27] (such as ANOVA followed by pairwise t-tests or Kruskal-Wallis followed by pairwise Mann-Whitney tests), and has been successfully applied in empirical software engineering [28], [29], [30]. We use the Tukey contrasts to compare all distributions pairwise.

To summarize the comparison results we use $\widetilde{\mathbf{T}}$-graphs [7] illustrated in Fig. 3. The $\widetilde{\mathbf{T}}$-graph in Fig. 3a indicates that the distribution represented by the node 0 is "greater" (shifted to the right) compared to the distribution represented by the node -1, and also compared to the distribution represented by the node -2. However, $\widetilde{\mathbf{T}}$ could not establish that the distribution represented by the node 0 is shifted to the right or to the left compared to the distribution represented by the node 1.

IV. Results

A. From Developers to Core Reviewers and back

Fig. 1 and Fig. 2 summarize the career paths of the full-time (upper) and non-full-time (lower) contributors. The size of a circle and the denominator in the fraction in the circle represent the total number of contributors that have *ever* belonged to the corresponding class. Fig. 1 represents every contributor only once, while in Fig. 2 if a contributor started as a reviewer, then became a core reviewer, and then again became a reviewer, this contributor will be counted twice when determining the denominator. The numerator inside the circle shows the size of the class in the most recent period. Transitions between classes are labeled with the numbers of contributors belonging to one class in a certain period, and to a different class in the subsequent period, as well as with the percentage of all contributors of the "source" class. Similarly to the circle sizes, in Fig. 2 if contributors takes a transition several times during their career, they will be counted twice. Incoming arrows with no source class correspond to contributors joining the class for the first time (or after a period of inactivity). Outgoing arrows with no target class represent those developers becoming inactive. Percentages on the remaining arrows indicate the outgoing contributors as percentage of all contributors that have *ever* been in the class. A *career path* is therefore a path in Fig. 2.

By comparing Fig. 1 and Fig. 2 we can observe that "re-entering" a class is not common. Indeed, 264 full-time core reviewers and 571 non-full-time core reviewers entered the core reviewers class 274 and 596 times respectively.

Inspecting Fig. 1 we see that while full-time contributors do not perform reviewing tasks, non-full-time developers are more likely to be reviewers. We conjecture that some of those non-full-time contributors are not formally employed by the OpenStack companies: indeed, everybody can register and become a reviewer to support a feature they like. Compared to full-time contributors, significantly higher share of non-full-time contributors of the same class leave OpenStack. Both for full-time and for the non-full-time developers migration

from or to the developers class is very limited, suggesting that if contributors do not start as reviewers they are unlikely to become reviewers later on. In both cases, there is more substantial migration between the reviewer and the core reviewer classes. However, also here substantial differences emerge between full-time and non-full-time contributors. Indeed, while ca. 38% of the non-full-time core reviewers stop being a core reviewer, only ca. 18% of the full-time core reviewers do so.

B. Easy Come, Easy Go

We start by investigating how fast one can become a core reviewer. Among 835 contributors that have been core reviewers at least once, 207 have became core reviewers immediately after a period of inactivity and another 261 became core reviewers after contributing for one period. 56% of the core reviewers thus become core reviewers in one period or less.

Focusing on the contributors that stop being core reviewers, the largest groups in Fig. 1 are non-full-time contributors that become regular reviewers and non-full-time contributors that become inactive. For both subgroups in more than 50% of the cases the decision to leave occurred after two periods or less, i.e., one year or less. We do not observe statistical differences between the durations of engagement (Mann-Whitney $p > 0.05$). Comparing full-time and non-full-time leavers of the core reviewers class, we observe that while as mentioned above 50% of the non-full-time contributors are no longer core reviewers after two periods, for the full-time contributors the 50% threshold is reached after three periods. However, the differences are not significant (Mann-Whitney $p > 0.05$).

C. Before and After Becoming Core Reviewer

In this section we focus on the differences in contributors' activity before and after becoming core reviewers. The corresponding $\widetilde{\mathbf{T}}$-graphs are summarized in Fig. 3. Time periods before the contributor becomes a core reviewer are denoted -2 and -1, the contributor becomes a core reviewer during period 0, the subsequent periods are denoted 1, 2 and 3.

Both for the full-time and for the non-full-time contributors the number of reviews contributed increases once the contributor becomes a core reviewer (Fig. 3a and 3b, respectively). No such uniformity can be observed for commits: while the number of commits authored by full-time contributors is higher once they became core reviewers (Fig. 3c, except for the number of commits carried out three periods after becoming the code reviewer) no such conclusion holds for non-full-time contributors (Fig. 3d). In all the four cases the transition period (0 in Fig. 3) has a highest level of activity.

V. Discussion

A. Career Path Hypotheses

Based on the case of OpenStack, we formulate the following hypotheses pertaining to career paths of contributors in modern industrial open source projects. Investigating those hypotheses should be a subject of a follow-up study.

H1 (cf. Section IV-A). Developers and (core) reviewers are different sub-populations within the contributors community and the movement between the sub-populations is rare. Movement between reviewers and core reviewers is more frequent.

H2a (cf. Section IV-B). The turnover of the core reviewers is high and occurs relatively rapidly. We would like to investigate whether this happens together with a change of affiliation or role in the same company.

H2b (cf. Section IV-B). Companies are interested in having core reviewers among their (full-time) staff and encourage the employees to obtain such a status. Indeed, the percentage of the full-time contributors progressing from the developer to the reviewer and further to the core reviewer is much higher than for the non-full-time contributors.

H3 (cf. Section IV-C). Becoming a core reviewer has a *signaling* effect in the community [31]. Indeed, we have found the highest levels of activity in the transition to become a core reviewer, so that it is beneficial for a career path to achieve such a position when looking for an (full-time) employment in the project. This would imply that core reviewers non-affiliated to companies will be more frequently hired by a company from the ecosystem than contributors of other classes.

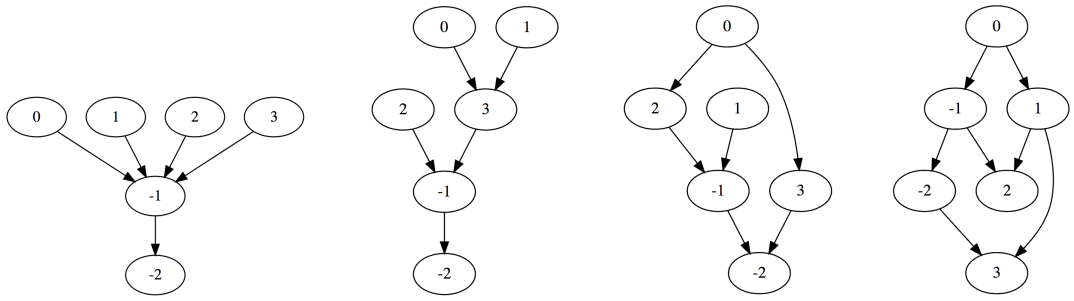
B. Threats to validity

As any empirical study, validity of our conclusions, i.e., the hypotheses above, might be threatened by several concerns. *Construct validity* depends on correctness of our operationalization of the full-time employment and career stages, and matching aliases corresponding to the same contributor. We opt for the "nine commits" heuristics [24] as an indication of full-time contribution since it has been designed and validated for Open Stack. Career stages might not be completely reflected by developer activities, however, we believe that code reviews provide are representative for career stages (Section II). Besides, the alias matching of Bitergia data had already satisfied the Open Stack Foundation as foundation members could contact Bitergia if mistakes were spotted. To ensure *internal validity* we have chosen a well-established statistical machinery [26] that has been successfully applied in the software engineering context in the past [28], [29], [30]. *External validity* depends on the completeness and correctness of the data dump as provided by Bitergia. These data have been used by Bitergia for the dashboards that are offered (and reviewed) by the OpenStack Foundation. Validity of the hypotheses for other modern multi-company Open Source projects is envisioned as a subject of a follow-up study.

VI. Conclusion

Motivated by the well-recognized importance of career perspectives for software engineers we conduct an exploratory case study of career paths within the OpenStack community. We operationalize career stages using the modern core review lens and distinguish between developers (non-reviewers), reviewers and core reviewers. Based on this case study we formulate four hypotheses about career paths of contributors in modern industrial open source projects.

As future work we consider confirming/rejecting the hypotheses first on other modern multi-company open source



(a) Reviews, full-time contributors (b) Reviews, non-full-time contributors (c) Commits, full-time contributors (d) Commits, non-full-time contributors

Fig. 3: Commits and reviews: periods before the contributor becomes a core reviewer are -2 and -1, the contributor becomes a core reviewer during period 0, the subsequent periods are 1, 2 and 3.

software projects such as OW2 and Netkit; and next on Open Source projects with a dominating company (e.g., Eclipse). In this way we hope to tease out the impact of presence of multiple competing and collaborating companies within an open source project on career advancement of the project contributors. In a complementary line of research we plan to explore perception of the career stages by the OpenStack contributors as well as managers of the companies involved.

REFERENCES

[1] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp, "Motivation in software engineering: A systematic literature review," *Information & Software Technology*, vol. 50, no. 9-10, pp. 860–878, 2008.

[2] A. C. C. França, T. B. Gouveia, P. C. F. Santos, C. A. Santana, and F. Q. B. da Silva, "Motivation in software engineering: A systematic review update," in *EASE*, April 2011, pp. 154–163.

[3] C.-G. Wu, J. H. Gerlach, and C. E. Young, "An empirical analysis of open source software developers: Motivations and continuance intentions," *Information & Management*, vol. 44, no. 3, pp. 253 – 262, 2007.

[4] J. A. Roberts, I.-H. Hann, and S. A. Slaughter, "Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the apache projects," *Management Science*, vol. 52, no. 7, pp. 984–999, 2006.

[5] C. Jensen and W. Scacchi, "Role migration and advancement processes in OSSD projects: A comparative case study," in *ICSE*, 2007, pp. 364–374.

[6] C. Jergensen, A. Sarma, and P. Wagstrom, "The onion patch: migration in open source ecosystems," in *FSE*. ACM, 2011, pp. 70–80.

[7] B. Vasilescu, A. Serebrenik, M. Goeminne, and T. Mens, "On the variation and specialisation of workload - A case study of the GNOME ecosystem community," *Empirical Software Engineering*, vol. 19, no. 4, pp. 955–1008, 2014.

[8] F. Fagerholm, A. S. Guinea, J. Borenstein, and J. Münch, "Onboarding in open source projects," *IEEE Software*, vol. 31, no. 6, pp. 54–61, 2014.

[9] I. Steinmacher, T. U. Conte, C. Treude, and M. A. Gerosa, "Overcoming open source project entry barriers with a portal for newcomers," in *ICSE*, 2016, pp. 273–284.

[10] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of Open Source Software development: Apache and Mozilla," *ACM Trans. Softw. Eng. Methodol.*, vol. 11, no. 3, pp. 309–346, Jul. 2002.

[11] R. G. Kula, A. E. Carmago Cruz, N. Yoshida, K. Hamasaki, K. Fujiwara, X. Yang, and H. Iida, "Using profiling metrics to categorise peer review types in the android project," in *ISSRE Workshops*, 2012, pp. 146–151.

[12] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in *ICSE*. IEEE Press, 2013, pp. 712–721.

[13] F. Ebert, F. Castor, N. Novielli, and A. Serebrenik, "Confusion detection in code reviews," in *ICSME*, 2017.

[14] N. Fenton and J. Bieman, *Software Metrics: A Rigorous and Practical Approach*, 3rd ed. CRC Press, Inc., 2014.

[15] G. Gousios, A. Zaidman, M.-A. Storey, and A. van Deursen, "Work practices and challenges in pull-based development: The integrator's perspective," in *ICSE*. IEEE Press, 2015, pp. 358–368.

[16] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2008.

[17] B. Flyvbjerg, *Five Misunderstandings about Case-Study Research*. Sage, 2007.

[18] J. Teixeira, "Understanding coopetition in the Open-Source arena: The cases of Webkit and OpenStack," in *International Symposium on Open Collaboration*. ACM, 2014, pp. 39.1–39.5.

[19] B. Lin, G. Robles, and A. Serebrenik, "Developer turnover in global, industrial Open Source projects: Insights from applying survival analysis," in *ICGSE*, 2017, pp. 66–75.

[20] G. Robles and J. M. González-Barahona, "Developer identification methods for integrated data from various sources," in *MSR*, 2005, pp. 1–5.

[21] E. Kouters, B. Vasilescu, A. Serebrenik, and M. G. J. van den Brand, "Who's who in GNOME: Using LSA to merge software repository identities," in *ICSM*, 2012, pp. 592–595.

[22] I. S. Wiese, J. T. da Silva, I. Steinmacher, C. Treude, and M. A. Gerosa, "Who is who in the mailing list? comparing six disambiguation heuristics to identify multiple addresses of a participant," in *ICSME*. IEEE Computer Society, 2016, pp. 345–355.

[23] M. Michlmayr, B. Fitzgerald, and K.-J. Stol, "Why and how should open source projects adopt time-based releases?" *IEEE Software*, vol. 32, no. 2, pp. 55–63, 2015.

[24] G. Robles, J. M. González-Barahona, C. Cervigón, A. Capiluppi, and D. Izquierdo-Cortazar, "Estimating development effort in free/open source software projects by mining software repositories: a case study of OpenStack," in *MSR*, 2014, pp. 222–231.

[25] A. Capiluppi and D. Izquierdo-Cortázar, "Effort estimation of FLOSS projects: A study of the Linux kernel," *Empirical Software Engineering*, vol. 18, no. 1, pp. 60–88, 2013.

[26] F. Konietschke, L. A. Hothorn, and E. Brunner, "Rank-based multiple test procedures and simultaneous confidence intervals," *Electronic Journal of Statistics*, vol. 6, pp. 738–759, 2012.

[27] K. R. Gabriel, "Simultaneous test procedures—some theory of multiple comparisons," *The Annals Mathematical Statistics*, vol. 40, no. 1, pp. 224–250, 1969.

[28] B. Vasilescu, A. Capiluppi, and A. Serebrenik, "Gender, representation and online participation: A quantitative study," *Interacting with Computers*, vol. 26, no. 5, pp. 488–511, 2014.

[29] Y. Yu, H. Wang, G. Yin, and T. Wang, "Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment?" *Inf. & Softw. Technology*, vol. 74, pp. 204–218, 2016.

[30] A. Swidan, A. Serebrenik, and F. Hermans, "How do Scratch programmers name variables and procedures?" in *SCAM*, 2017.

[31] B. Vasilescu, A. Serebrenik, P. T. Devanbu, and V. Filkov, "How social Q&A sites are changing knowledge sharing in Open Source Software communities," in *CSCW*, 2014, pp. 342–354.