ICSE: U: Adding Sparkle to Social Coding: An Empirical Study of Repository Badges in the *npm* Ecosystem

Asher Trockman
University of Evansville, USA
asher.trockman@gmail.com

1 PROBLEM AND MOTIVATION

Open source is everywhere, acting as "digital infrastructure" for everything from social media, to medical records, to scientific computing [17]; it is relied upon by governments, companies, startups, and independent users. The economic value of open source has been measured in the billions [15, 19]; clearly, the sustainability of open source software is of utmost importance.

Open source has changed over the past decade. This is in large part due to the rise of the social coding website GitHub, the standard place to collaborate on code. GitHub hosts millions of repositories of libraries and tools, which developers reuse liberally [18], creating complex and often fragile networks of interdependencies [7]. Simultaneously, the demand for open source software is increasing, and the supply of developer time and effort cannot always keep up. These factors have resulted in notable failures of open source software, *e.g.*, the Heartbleed and left-pad incidents [17].

A significant change is the unprecedented *transparency* provided by social coding platforms like GitHub [13, 14], where user profile pages display information on one's contributions, the development history of projects is archived and publicly accessible, and repository pages provide information on a project's social standing (*e.g.*, through *stars* and *watchers*). This transparency can enhance collaboration and coordination [14]. Using such visible cues, known as *signals* [37], found on profile and repository pages, developers can better manage their projects and dependencies, communicate more efficiently, become informed about action items requiring their attention, learn, socialize, and form impressions about each other's coding ability and personal characteristics [14, 28, 29, 43].

Our work studies the effects of transparency on open source development with a focus on decision making under uncertainty. Contemporary software development is characterized by increased reuse and speed, and developers must quickly make decisions that have sustainability consequences: users must find which libraries to adopt as dependencies, contributors must decide which projects to contribute to, and project maintainers must allocate their time to enhancing their projects and communicating their underlying qualities to these audiences, often while multitasking [42]. One mechanism for closing the gap between supply and demand in open source is the perception and decision-making of individual developers [17], which is supported by transparency. All of these decisions can impact the sustainability of open source software, as they can influence the balance of supply and demand for contributor effort.

As part of the larger research agenda outlined above, we focus on one particular transparency mechanism: *repository badges*, images such as 

, which are embedded in projects' README files and are often dynamically generated. Badges can be seen as signaling mechanisms, increasing transparency by quickly providing insights into otherwise hard-to-see project qualities such as test suite quality, openness to contributions, and dependency management practices. Badges present a unique opportunity to do a natural experiment, as thousands of projects have adopted them. Through large-scale statistical analyses, we can discern the effects of badges and later propose interventions based on our discoveries, which could improve decision making and sustainability.

We explore two main research questions regarding badges: First, we explore the phenomenon quantitatively and qualitatively, and ask (**RQ**$_1$) *What are the most common badges and what does displaying them intend to signal?* Second, we analyze whether badges actually signal what developers expect, and ask (**RQ**$_2$) *To what degree do badges correlate with qualities that developers expect?* To this end, we perform a large-scale mixed-methods empirical study of badges in the *npm* ecosystem, a large and vibrant open-source ecosystem for JavaScript with documented interdependency-related coordination challenges [7], wherein many badges originated.

If we understood which badges were reliable signals, developers could make more informed choices about which of these projects to depend on, more effectively evaluating security, well-testedness, and availability of support. Contributors could better determine which projects follow suitable development practices and are open to new contributions. Project maintainers could be more deliberate when selecting badges to display, providing evidence for their good practices. The developers of badge-providing services could design badges that provide a better assessment of long-term adherence to quality standards. In summary, badges are a potentially impactful feature in transparent, social coding environments; our study provides new understanding of their value and effects. More broadly, better signals could highlight and incentivize developer effort on pieces of infrastructure that need it most.

Acknowledgments. The work was done in collaboration with researchers at Carnegie Mellon in their REUSE program, and with researchers at UC Davis in later related work.

2 BACKGROUND & RELATED WORK

Related work. Research has confirmed that signals in online profiles are used as indicators of expertise and commitment [13, 28, 34, 40, 41], *e.g.*, Stack Overflow reputation score and GitHub followers. We expect that badges may have a similar effect. Studies have looked into specific practices in software ecosystems, *e.g.*, dependency updates [2, 12, 21, 25, 31], static analysis [3, 46], and continuous integration [22, 44, 47]. In particular, concurrently with this study, Mirhosseini *et al.* investigated the efficacy of automatic pull requests versus badges in convincing open source developers

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden
2018. ACM ISBN 978-1-4503-5663-3/18/05…$15.00
https://doi.org/10.1145/3183440.3190335

to upgrade dependencies; while both are significant, the latter has a greater effect [31]. In contrast, our study is unique in that it views badges more broadly as *signals* beyond individual tools and practices; we also investigate many temporal correlates of badge adoption through time series analysis in a quasi-experimental design.

Theoretical Framework. We frame our study in the context of *signaling theory* [38], which is widely applied to selection scenarios in disciplines ranging from economics [37] to biology [45]. The classic example in economics is job market candidates signaling their ability to employers, which is otherwise hard to observe, by holding a degree from a prestigious institution [37]. Similarly, selection scenarios occur routinely in open source, *e.g.*, choosing which libraries to depend on [7], repositories to watch [35], developers to follow [6, 26] or hire [9, 27], and projects to contribute to [5, 10].

Signals are observable pieces of information that indicate a hidden quality of the signaler. We argue that badges are *signals* because they make certain information about a project's code base or practices transparent, thereby reducing *information asymmetry* between maintainers and their users and contributors; badges make it easy to observe project properties that are otherwise hidden. However, badges vary widely in production cost and thus reliability: Some badges are *assessment signals*, which are expensive to produce, *e.g.*, being the result of a third-party analysis of the codebase; *e.g.*, David's dependencies up to date reports whether dependencies are up-to-date and secure, and Coveralls' coverage 1% reports a test coverage metric. Others are *conventional signals*, which indicate qualities that are easy to look up elsewhere, *e.g.*, license BSD, npm v1.1.0, and PRs welcome.

3 APPROACH & UNIQUENESS

Our goal is to evaluate the reliability and effects of badges as signals by *quantitatively* assessing *signal fit*, *i.e.*, the extent that signals correspond to the desirable, unobservable quality of the signaler. According to theory, assessment signals should be the most reliable.

Survey. First, to better understand what maintainers intend to signal with badges and how developers perceive those badges, we conducted a survey of *npm* maintainers and contributors, sending 580 emails and receiving 32 maintainer and 57 contributor responses. Both surveys required specific badges and their expected, perhaps otherwise unobservable, associated qualities to be mentioned. For two examples: 84% of maintainers said that they use quality assurance badges like build passing and coverage 94% to signal code and development quality, *e.g.*, that they "have tests and run them regularly"; and 26% of responses mentioned dependency management badges, which signal attention to updates and security patches. A majority, 88% of respondents, agreed that "the presence of badges is an indicator of project quality."

Data mining. To study the adoption and the effects associated with badges quantitatively, we mined a longitudinal data set of 294,941 *npm* packages. We collected metadata on downloads and releases from *npm* and project history, such as test suite size, historical dependencies, and commit counts from GitHub using custom data mining tools. Badges and their adoption dates were extracted from the git history of each repository's README file. We iteratively devised regular expressions and keywords for badge identification and classification. We found 88 distinct types of badges and split them into six categories: quality assurance, dependency management, information, popularity, support, and other (see Table 1).

Table 1: Some categories of badges present in our data.

Badge	Name	Description	Adoption	ST
Quality Assurance				
build passing	Travis CI	Build status	92789 (31.5%)	A
coverage 53%	Coveralls	Test coverage	17603 (6.0%)	A
code climate 4.0	CodeClimate	Coverage & static analysis	6652 (2.3%)	A
Dependency Management				
dependencies up to date	David DM	Version tracking	23601 (8.0%)	A
dependencies out of date	Gemnasium	Version tracking	2851 (1.0%)	A
Greenkeeper enabled	Greenkeeper	Version tracking	1599 (0.5%)	A
Information				
release v2.1.1	Version	*npm*/GitHub version	64200 (21.8%)	L
license BSD	License	License information	6250 (2.1%)	S
code style standard	JS Standard	Coding style	5299 (1.8%)	S
Popularity				
downloads 654/month	Downloads	*npm* download statistics	15552 (5.3%)	L
cdnjs v3.2.1	cdnjs	Host of popular libraries	1902 (0.6%)	L
Follow 350	Twitter	Twitter link and stats	810 (0.3%)	S
Support				
gitter join chat	Gitter	Chat & collaboration	4786 (1.6%)	S
issue resolution 3 h	GitHub Issues	Issue statistics	1213 (0.4%)	L
slack 6/160	Slack	Chat & collaboration	688 (0.2%)	S
Other				
Donate	Donation link	PayPal, Patreon, ...	1632 (0.6%)	S
tips $3.64/week	Donation stats	Gratipay, Gittip, ...	919 (0.3%)	L
ember observer 8 / 10	Ember Observer	Reviews and scoring	474 (0.2%)	A

ST (signal type): A—assessment signal based on nontrivial analysis or aggregation; L—lookup of readily available information; S—static statement of information

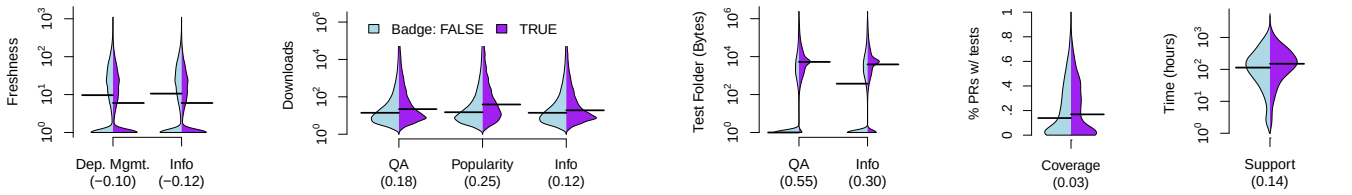
Data analysis. To check hypotheses about developer perceptions from the survey, we follow three complementary steps, each analyzing the correlation of badges with a quality at a deeper level:

Step 1: Correlation. We explore if badges are reliable signals of certain qualities without concern for causal relationships, confounds, or historical trends; *e.g.*, which badges correlate with larger test suites? We compare distributions (see Fig. 1) using the WMW test and report Cliff's delta for effect size.

Step 2: Additional information. Here we investigate correlation while controlling for other visible indicators of project quality (*e.g.*, stars, dependents, age); *e.g.*, does a project with a quality assurance badge have a larger test suite, other factors held equal? We compare a base regression model without badges to a full one with badges.

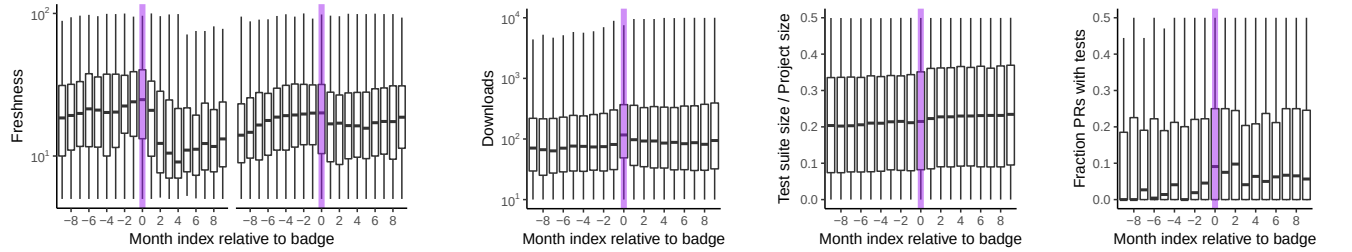
Step 3: Longitudinal analysis. We use an *interrupted time series design* [11], measuring a quality at 19 monthly intervals, with the middle month being that in which a badge is adopted (Fig. 2). The change in the trend after the adoption reveals whether badges are indicative of lasting changes to development practices, *e.g.*, do projects typically write more tests after adopting a quality assurance badge?

Uniqueness. Many studies have looked into specific practices in software ecosystems, including communication [4, 20, 23, 36], change planning [7, 8, 16, 32], dependency updates [2, 12, 21, 25, 31], static analysis [3, 46], testing and continuous integration [22, 44, 47], and many others; badges and their underlying tools are also ecosystem-level practices. In contrast to prior work, however, we specifically take a broader view on badges as *signals* beyond individual tools and practices. Moreover, while effects associated with adopting these tools have been studied by prior work [3, 22, 31, 44, 46, 47], our approach is unique in that we focus on the *signaling*



(a) Dependency freshness **(b) Popularity** **(c) Test suite size** **(d) PRs with tests** **(e) Issue latency**

Figure 1: Distributions of response variables w/o and w/ badges. Horizontal lines depict medians. Cliff's delta below each plot.



(a) Monthly freshness scores, rel. to dependency-manager (left) and information badges (right). **(b) Monthly downloads, relative to first badge.** **(c) Ratio test suite size / package size, rel. to QA badge.** **(d) Fraction PRs with tests, relative to QA badge.**

Figure 2: Trends in response variables before and after badge adoption (month highlighted).

dimension added by repository badges to these and other tools, previously overlooked in the software engineering signaling literature (*e.g.*, [13, 28, 34, 40, 41]).

We conduct a unique mixed-methods study that combines insights from a survey of open-source developers with sophisticated statistical analyses of a large-scale, diverse data set, all framed within a strong, well-established economic theory. This triangulation lends confidence to our result that assessment signal badges are more reliable than conventional signals. Our study is the first to investigate signaling in open source software quantitatively and over time, making use of the powerful interrupted time series quasi-experimental design. It paves the way for future experiments on the efficacy of signals on social coding websites and is of practical importance to open source developers.

4 RESULTS

Badge adoption. We found that 46% of packages have adopted at least one badge; only few are broadly adopted, with the most common being Travis CI build passing (see Table 1). They are adopted in groups and are not frequently updated or removed.

Signals of updated dependencies. To clearly illustrate our three-step process, we first focus on dependency management badges. As the response, we created a metric of dependency up-to-dateness, or *freshness*, based on previous work [12]. A lower score is better; zero means the project's dependencies are entirely up-to-date.

Based on our survey, we hypothesize that dependency management badges (*e.g.*, dependencies up to date), which indicate whether a project's dependencies are outdated or insecure, are correlated with dependency freshness. Furthermore, since information badges (*e.g.*, release v2.1.1) primarily provide convenient links and are not associated with an assessment of a quality, we hypothesize that they do not systematically influence freshness.

Correlation. Packages with dependency management badges tend to have fresher dependencies than those without. Surprisingly, we see the same effect for those with just information badges (Fig. 1a).

Additional information. To test if the presence of these badges is associated with a deeper indication of freshness beyond other readily available signals, we fit a hurdle regression: a logistic regression to model whether *freshness* = 0 and a linear regression to model the level of freshness. This approach is necessary since 37% of packages with dependencies have up-to-date (*freshness* = 0) dependencies. For the logistic regression, we found that the odds of having up-to-date dependencies increases by 27% if a project has a dependency management badge. Surprisingly, information badges are correlated with a similar increase of 17%. We see similar results for the linear regression.

Longitudinal analysis. We collect a sample of 1,761 packages that have 9 months of history before and after the adoption of the first badge and at least one month with *freshness* ≠ 0 in that time frame. In Fig. 2a, a trend is clearly visible, which is supported by a statistical model. The adoption of *any* badge is correlated with a strong improvement in freshness. The freshness slightly decays over time, *i.e.*, the change in practices does not last. As hypothesized, the adoption of a dependency management badge is associated with a longer-lasting effect on freshness than other badges. The additional effect of information badges on the decay is negligible.

Discussion. The results from the three preceding steps support our hypothesis that dependency management badges are reliable signals of practices that lead to fresher dependencies. Though dependency management badges are correlated with a stronger and longer-lived effect, the effect is is not exclusive to dependency management badges: we speculate that any maintenance task involving the addition of badges might involve other project cleanup efforts. Developers and contributors can use this information to select projects that are more likely to be up-to-date and secure.

Signals of test suite quality. Based on our survey, we expect that the adoption of quality assurance badges (such as  and ) correlates with an increase in test suite quality, operationalized as the size of the test suite, with at most a marginal effect from information badges. We follow our 3-step analysis.

Correlation. The distribution of test suite sizes is bimodal, with 39.3% of packages having no test code; packages with quality-assurance badges almost always have tests (93.5%). Among packages with tests, those with badges tend to have larger test suites; surprisingly, the difference is similar for information badges (see Fig. 1c).

Additional information. As with dependencies, we fit a hurdle regression, modeling separately the likelihood of having any tests (logistic regression) and the test suite size for packages with tests (negative binomial regression). Both models show a large positive effect for the presence of quality assurance badges: the odds of having tests increases 18× with the badges, and packages with existing tests are expected to have 18.3% larger test suites, other variables held constant. As expected, information badges have a marginal effect, though they interact positively with quality assurance badges.

Longitudinal analysis. To avoid technical problems with the bimodal distribution, we only investigate how the first badge adoption correlates with the growth of an *existing* test suite. We assemble a longitudinal sample of 2,855 packages with at least 9 months of history with non-empty test suites before and after the badge adoption month. Though it is barely visible in the plotted data (Fig. 2c), there is a small but statistically significant positive shift in test suite size at the adoption date, but no change in slope afterwards. That is, adopting quality assurance badges coincides with an improvement in the test suite, but not with a lasting change to testing practices. Information badges are correlated with a much smaller effect.

Discussion. As expected by the surveyed maintainers, all three steps indicate that quality-assurance badges are good signals for the presence of tests, though they are a weaker signal for the size of the test suite. We see a marginal change for information badges, which may be explained project cleanup efforts coincident with changes to the README file.

Summary of other results. Using the same method as above, we also found that build status and coverage badges encourage external contributors to include more tests in pull requests (Fig. 1d, 2d). We found that popularity badges correlate with gains in downloads, though the effect does not persist (Fig. 1b, 2b). Badges are inextricably tied to third-party services, yet we isolated the effects of badges by comparing with projects that adopted Travis CI without the associated badge; projects with the badge are more likely to have passing builds than those without, and the badge adoption correlates with a larger increase in test suite size than merely adopting Travis CI.

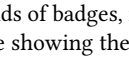
Threats to validity. *External validity.* As typical for a survey, our sample may suffer from selection bias. One should be careful when generalizing the results beyond the studied corpus of *npm* packages, as the *npm* community has certain characteristics that may differ in other communities; *e.g.*, Python and Java developers may adopt different tools and innovations at a different pace, and not all package repositories show READMEs and badges as prominently as *npm*. The results may differ outside of open source, *e.g.*, when using badges in corporate teams.

Imperfect measures. Our operationalized measures cannot fully capture the underlying qualities mentioned in the survey, but we expect a strong correlation on average given our large data set. One must be careful when generalizing beyond the studied measures.

5 CONTRIBUTIONS

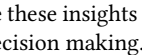
We studied repository badges, a new phenomenon in social coding environments like GitHub with previously unknown effects.

Research questions. By exploring the most common types of badges and their intended signals ($\mathbf{RQ}_1$), we found a diversity of badges and signals; some are merely static displays of existing information, *i.e.*, conventional signals, and others aggregate hard-to-observe information like build status, up-to-dateness of dependencies, and test coverage, *i.e.*, assessment signals. Our survey revealed that package maintainers have clear signaling intentions, and many developers interpret badges when evaluating packages. Exploring the fit of the signals to the underlying qualities hypothesized by our survey participants ($\mathbf{RQ}_2$), we found that packages with badges tend to have more of the quality they intend to signal, a small effect remaining even when controlling for other visible signals. Correlations are particularly strong for *assessment signals*, or badges that test an underlying non-trivial quality, *e.g.*, quality assurance and dependency management badges, rather than just stating intentions, *e.g.*, information badges. Badges tend to correlate with a positive increase in the qualities they signal around the time of their adoption. For assessment signals, this effect tends to be stronger and longer-lived, sometimes suggesting a lasting change to development practices.

Implications for practitioners. Based on our results, *package maintainers* can make more deliberate choices about badges, *e.g.*, by favoring assessment signals. For example, Slack offers multiple kinds of badges, including one inviting users to join () and one showing the number of active users (). In practice, most maintainers adopt the former (conventional signal) rather than the latter (assessment signal). Our results indicate that maintainers who have the choice and are serious about signaling their dedication should adopt the assessment signal. *Service developers* can design badges more carefully by providing an assessment signal based on some analysis of past conformance. *Package users and contributors* can better decide which badges to use as indicators of underlying practices and as starting points to investigate deeper qualities.

Impact. To date, for example, the original study has been highlighted by *npm* [30] and cited by open source developers [33].

Future work. We have found evidence that badges are reliable indicators of the usage of hard-to-observe tools and development practices; this enables more empirical software engineering research, *e.g.*, on the outcomes of using such tools [24]. Further, detecting badges could enable social network analysis to find factors influencing the spread of development practices across GitHub.

Our study inspires the design (or redesign) of badges that are stronger assessment signals, though more work remains, *e.g.*, to see how the *value* on the badge matters for developer decision making, *e.g.*,  versus  . Future experiments may leverage these insights to create new signals and evaluate their impact on decision making.

More studies are needed to determine how badges influence project attractiveness and the decision to contribute [1]. Broadly, we

think that future studies to gain a better understanding of signaling in open source could help to redirect scarce resources to crucial infrastructure; badges could be designed which steer contributors towards projects in need.

Conclusion. Overall: signals mostly reliable. More details can be found in a related publication [39].

REFERENCES

[1] Anonymous Authors. 2019. The Signals that Potential Contributors Look for When Choosing Open-source Projects (under review).

[2] Gabriele Bavota, Gerardo Canfora, Massimiliano Di Penta, Rocco Oliveto, and Sebastiano Panichella. 2015. How the Apache community upgrades dependencies: An evolutionary study. *Empirical Software Engineering* 20, 5 (2015), 1275–1317.

[3] Moritz Beller, Radjino Bholanath, Shane McIntosh, and Andy Zaidman. 2016. Analyzing the state of static analysis: A large-scale evaluation in open source software. In *Proc. Int'l Conf. Software Analysis, Evolution and Reengineering (SANER)*, Vol. 1. IEEE, 470–481.

[4] Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, and Anand Swaminathan. 2006. Mining email social networks. In *Proc. Working Conf. Mining Software Repositories (MSR)*. ACM, 137–143.

[5] Christian Bird, Alex Gourley, Prem Devanbu, Anand Swaminathan, and Greta Hsu. 2007. Open borders? Immigration in open source projects. In *Proc. Working Conf. Mining Software Repositories (MSR)*. IEEE, 6.

[6] Kelly Blincoe, Jyoti Sheoran, Sean Goggins, Eva Petakovic, and Daniela Damian. 2016. Understanding the popular users: Following, affiliation influence and leadership on GitHub. *Information and Software Technology* 70 (2016), 30–39.

[7] Christopher Bogart, Christian Kästner, James Herbsleb, and Ferdian Thung. 2016. How to break an API: Cost negotiation and community values in three software ecosystems. In *Proc. Int'l Symp. Foundations of Software Engineering (FSE)*. ACM, 109–120.

[8] John Businge, Alexander Serebrenik, and Mark GJ van den Brand. 2015. Eclipse API usage: the good and the bad. *Software Quality Journal* 23, 1 (2015), 107–141.

[9] Andrea Capiluppi, Alexander Serebrenik, and Leif Singer. 2013. Assessing technical candidates on the social web. *IEEE Software* 30, 1 (2013), 45–51.

[10] Casey Casalnuovo, Bogdan Vasilescu, Premkumar Devanbu, and Vladimir Filkov. 2015. Developer onboarding in GitHub: the role of prior social links and language experience. In *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)*. ACM, 817–828.

[11] Thomas D Cook, Donald Thomas Campbell, and Arles Day. 1979. *Quasi-experimentation: Design & analysis issues for field settings*. Vol. 351. Houghton Mifflin Boston.

[12] Joël Cox, Eric Bouwers, Marko van Eekelen, and Joost Visser. 2015. Measuring Dependency Freshness in Software Systems. In *Proc. Int'l Conf. Software Engineering, Volume 2*. IEEE Press, 109–118.

[13] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proc. Conf. Computer Supported Cooperative Work (CSCW)*. ACM, 1277–1286.

[14] Laura Dabbish, Colleen Stuart, Jason Tsay, and James Herbsleb. 2013. Leveraging transparency. *IEEE Software* 30, 1 (2013), 37–43.

[15] Carlo Daffara. 2012. Estimating the economic contribution of open source software to the European economy. In *The First Openforum Academy Conference Proceedings*.

[16] Alexandre Decan, Tom Mens, and Maëlick Claes. 2017. An empirical comparison of dependency issues in OSS packaging ecosystems. In *Proc. Int'l Conf. Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2–12.

[17] Nadia Eghbal. 2016. *Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure*. Ford Foundation.

[18] Mohammad Gharehyazie, Baishakhi Ray, and Vladimir Filkov. 2017. Some from here, some from there: cross-project code reuse in GitHub. In *Proc. Working Conf. Mining Software Repositories (MSR)*. IEEE, 291–301.

[19] Shane Greenstein and Frank Nagle. 2014. Digital dark matter and the economic contribution of Apache. *Research Policy* 43, 4 (2014), 623–631.

[20] Anja Guzzi, Alberto Bacchelli, Michele Lanza, Martin Pinzger, and Arie Van Deursen. 2013. Communication in open source software development mailing lists. In *Proc. Working Conf. Mining Software Repositories (MSR)*. IEEE, 277–286.

[21] Joseph Hejderup. 2015. *In dependencies we trust: How vulnerable are dependencies in software modules?* Master's thesis. TU Delft, The Netherlands.

[22] Michael Hilton, Timothy Tunnell, Kai Huang, Darko Marinov, and Danny Dig. 2016. Usage, Costs, and Benefits of Continuous Integration in Open-source Projects. In *Proc. Int'l Conf. Automated Software Engineering (ASE)*. ACM, 426–437.

[23] 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 *Proc. Int'l Conf. Software Engineering (ICSE)*. IEEE Press, 164–174.

[24] David Kavaler, Asher Trockman, Bogdan Vasilescu, and Vladimir Filkov. Tool Choice Matters: JavaScript Quality Assurance Tools and Usage Outcomes in GitHub Projects. *Management* 20 (????), 763.

[25] Raula Gaikovina Kula, Daniel M German, Ali Ouni, Takashi Ishio, and Katsuro Inoue. 2017. Do developers update their library dependencies? *Empirical Software Engineering* (2017), 1–34.

[26] Michael J Lee, Bruce Ferwerda, Junghong Choi, Jungpil Hahn, Jae Yun Moon, and Jinwoo Kim. 2013. GitHub developers use rockstars to overcome overflow of news. In *Proc. Conf. Human Factors in Computing Systems (CHI), Extended Abstracts*. ACM, 133–138.

[27] Jennifer Marlow and Laura Dabbish. 2013. Activity traces and signals in software developer recruitment and hiring. In *Proc. ACM Conference on Computer Supported Cooperative Work (CSCW)*. ACM, 145–156.

[28] Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. 2013. Impression formation in online peer production: activity traces and personal profiles in GitHub. In *Proc. Conf. Computer Supported Cooperative Work (CSCW)*. ACM, 117–128.

[29] Vishal Midha and Prashant Palvia. 2012. Factors affecting the success of Open Source Software. *Journal of Systems and Software* 85, 4 (2012), 895–905.

[30] Inc. npm. 2018. npm weekly #138: Take a look at our latest job listings, learn what badges do, ... https://medium.com/npm-inc/npm-weekly-138-take-a-look-at-our-latest-job-listing-learn-what-badges-do-and-see-npm-ci-work-d74a514fa3d6. (March 2018).

[31] Chris Parnin et al. 2017. Can Automated Pull Requests Encourage Software Developers to Upgrade Out-of-Date Dependencies?. In *Proc. Int'l Conf. Automated Software Engineering (ASE)*. to appear.

[32] Steven Raemaekers, Arie van Deursen, and Joost Visser. 2014. Semantic Versioning versus Breaking Changes: A Study of the Maven Repository. In *Proc. Int'l Working Conf. Source Code Analysis and Manipulation (SCAM)*. IEEE Computer Society, 215–224.

[33] Gabriel Schulhof. 2018. Badges for N-API add-ons. https://github.com/nodejs/admin/issues/221. (August 2018).

[34] N Sadat Shami, Kate Ehrlich, Geri Gay, and Jeffrey T Hancock. 2009. Making sense of strangers' expertise from signals in digital artifacts. In *Proc. Conf. Human Factors in Computing Systems (CHI)*. ACM, 69–78.

[35] Jyoti Sheoran, Kelly Blincoe, Eirini Kalliamvakou, Daniela Damian, and Jordan Ell. 2014. Understanding watchers on GitHub. In *Proc. Working Conf. Mining Software Repositories (MSR)*. ACM, 336–339.

[36] Leif Singer, Fernando Figueira Filho, and Margaret-Anne Storey. 2014. Software engineering at the speed of light: How developers stay current using Twitter. In *Proc. Int'l Conf. Software Engineering (ICSE)*. ACM, 211–221.

[37] Michael Spence. 1973. Job market signaling. *The Quarterly Journal of Economics* 87, 3 (1973), 355–374.

[38] Michael Spence. 2002. Signaling in retrospect and the informational structure of markets. *The American Economic Review* 92, 3 (2002), 434–459.

[39] Asher Trockman, Shurui Zhou, Christian Kästner, and Bogdan Vasilescu. 2018. Adding Sparkle to Social Coding: An Empirical Study of Repository Badges in the npm Ecosystem. In *Proceedings of the 40th International Conference on Software Engineering (ICSE)*. ACM Press, New York, NY.

[40] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Influence of social and technical factors for evaluating contribution in GitHub. In *Proc. Int'l Conf. Software Engineering (ICSE)*. ACM, 356–366.

[41] Jason Tsay, Laura Dabbish, and James D Herbsleb. 2013. Social media in transparent work environments. In *Proc. International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 65–72.

[42] Bogdan Vasilescu, Kelly Blincoe, Qi Xuan, Casey Casalnuovo, Daniela Damian, Premkumar Devanbu, and Vladimir Filkov. 2016. The sky is not the limit: multitasking across GitHub projects. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 994–1005.

[43] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. 2015. Perceptions of diversity on GitHub: A user survey. In *Proc. Int'l Workshop Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 50–56.

[44] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. 2015. Quality and Productivity Outcomes Relating to Continuous Integration in GitHub. In *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE) (ESEC/FSE)*. IEEE, 805–816.

[45] Amotz Zahavi. 1975. Mate selection—a selection for a handicap. *Journal of Theoretical Biology* 53, 1 (1975), 205–214.

[46] Fiorella Zampetti, Simone Scalabrino, Rocco Oliveto, Gerardo Canfora, and Massimiliano Di Penta. 2017. How open source projects use static code analysis tools in continuous integration pipelines. In *Proc. Working Conf. Mining Software Repositories (MSR)*. IEEE, 334–344.

[47] Yangyang Zhao, Yuming Zhou, Alexander Serebrenik, Vladimir Filkov, and Bogdan Vasilescu. 2016. The Impact of Continuous Integration on Other Software Development Practices: A Large-Scale Empirical Study. In *Proc. Int'l Conf. Automated Software Engineering (ASE)*. IEEE.