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 INTRODUCTION AND MOTIVATION

Contemporary software development is characterized by increased reuse and speed. Open source software forges such as GitHub host millions of repositories of libraries and tools, which developers reuse liberally [6], creating complex and often fragile networks of interdependencies [1]. Hence, developers must make more decisions at a higher speed, finding which libraries to depend on and which projects to contribute to. This decision making process is supported by the *transparency* provided by social coding platforms like GitHub [4, 5], where user profile pages display information on a one's contributions, and repository pages provide information on a project's social standing (*e.g.*, through *stars* and *watchers*).

Using such visible cues, known as *signals* [9], 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, personal characteristics, and interpersonal skills [5, 7, 8, 11].

We focus on *repository badges*, images such as 

, which are embedded in projects' README files and 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, adherence to coding standards, dependency management practices, and openness to contributions.

We investigate which qualities maintainers intend to signal with badges and how well badges corrrelate with those qualities. 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 [1], wherein many badges originated.

The *npm* ecosystem contains a vast number of competing projects which provide similar functionality. 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 welcoming to new contributors. Project maintainers could be more deliberate when selecting badges to display, highlighting and 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.

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05…$15.00
https://doi.org/10.1145/3183440.3190335

2 APPROACH

Survey. 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 the qualities expected to be associated with them to be mentioned. A majority, 88% of respondents, agreed that "the presence of badges in general is an indicator of project quality."

Data mining. To study the adoption and the effects associated with badges, 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. 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.

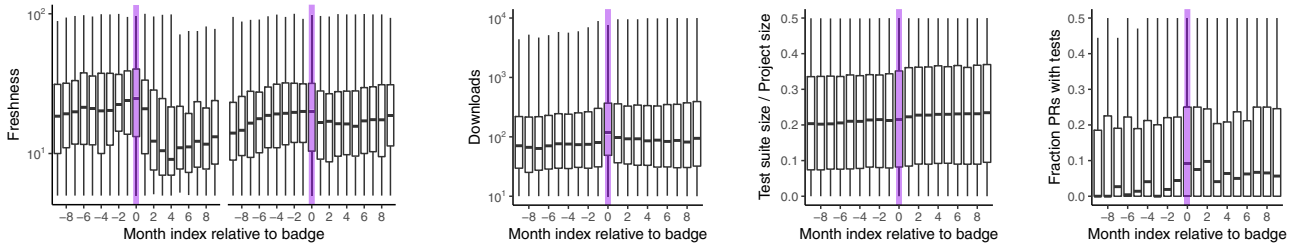
We found 88 distinct types of badges and split them into six categories: *quality assurance* (*e.g.*, continuous integration build status), *dependency management* (version and vulnerability tracking), *information* (*e.g.*, latest release, coding style), *popularity* (*e.g.*, downloads, CDN availability), *support* (*e.g.*, chat, issue statistics), and *other*.

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:

Correlation. We explore whether badges are reliable signals of certain qualities without concern for causal relationships, confounds, or historical trends; *e.g.*, do quality assurance badges correlate with larger test suites (in bytes)?

Additional information. Here we investigate correlation while controlling for other visible indicators of project quality (*e.g.*, stars and dependents); 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.

Longitudinal analysis. We use a *time-series regression discontinuity design* [2], measuring a quality at 19 monthly intervals, with the middle month being that in which a badge is adopted. The change in the trend after the adoption can reveal whether badges are indicative of lasting changes to development practices.



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

3 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. They are adopted in groups and are not frequently updated or removed.

Signals of updated dependencies. To clearly illustrate our three-step process, we focus on dependency management badges. As the response, we created a metric of dependency up-to-dateness, or *freshness*, based on previous work [3]. 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 packages with just information badges.

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 $freshness \neq 0$ in that time frame. In Fig. 1a, 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.

Summary of other results. Using the same method as above, we also found that build status and coverage badges correlate with increased test suite sizes (Fig. 1c) and encourage external contributors to include more tests in pull requests (Fig. 1d). We found that popularity badges correlate with gains in downloads, though the effect does not persist (Fig. 1b). 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. 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.

4 CONTRIBUTIONS

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, 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. *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.

Overall: signals mostly reliable. More details can be found in a related publication [10].

REFERENCES

[1] 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.

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

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

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

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

[10] 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.

[11] 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.