GitQ- Towards Using Badges as Visual Cues for GitHub Projects

Akhila Sri Manasa Venigalla*
Kowndinya Boyalakuntla*
Sridhar Chimalakonda
Research in Intelligent Software & Human Analytics (RISHA) Lab
Department of Computer Science and Engineering
Indian Institute of Technology Tirupati, Tirupati, India
{cs19d504,cs17b032,ch}@iittp.ac.in

ABSTRACT

GitHub hosts millions of software repositories, facilitating developers to contribute to many projects in multiple ways. Most of the information about the repositories is text-based in the form of stars, forks, commits, and so on. However, developers willing to contribute to projects on GitHub often find it challenging to select appropriate projects to contribute to or reuse due to the large number of repositories present on GitHub. Further, obtaining this required information often becomes a tedious process, as one has to carefully mine information hidden inside the repository. To alleviate the effort intensive mining procedures, researchers have proposed npm-badges to outline information relating to build status of a project. However, these badges are static and limit their usage to package dependency and build details. Adding visual cues such as badges, `Inherited Classes 2943` to the repositories might reduce the search space for developers. Hence, we present *GitQ*, to automatically augment GitHub repositories with badges representing information about source code and project maintenance. Presenting *GitQ* as a browser plugin to GitHub could make it easily accessible to developers using GitHub. *GitQ* is evaluated with 15 developers based on the UTAUT model to understand developer perception towards its usefulness. We observed that 11 out of 15 developers perceived *GitQ* to be useful in identifying the right set of repositories using visual cues such as `Methods 17367100` generated by *GitQ*. The source code and tool are available for download on GitHub at https://github.com/gitq-for-github/plugin, and the demo can be found at https://youtu.be/c0yohmIat3A.

CCS CONCEPTS

• **Software and its engineering** → **Software libraries and repositories**; **Software notations and tools**.

KEYWORDS

software quality, quality badges, ck metrics, github repositories

*Equal contribution

1 INTRODUCTION

GitHub is extensively used among the available open-source code sharing platforms and currently hosts more than 72 million users[1], with nearly 43 million public repositories[2]. GitHub facilitates collaborative coding [17] and enables users to identify their peer developers' goals and suggest updates through pull requests, issues and commit messages [8, 16]. Most developers *fork* a repository to reuse its source code if they find repositories with similar requirements [10]. Readme-files and wikis of a repository are expected to provide information about the repository to the users [1, 28]. However, it has been observed that the existing readme-files do not suffice to identify appropriate repositories [13]. Even though it is effort-intensive, developers do not have a choice but to browse and analyze source code manually to understand about a repository. Researchers have developed several tools to assist developers in understanding GitHub repositories [5, 12, 16]. These tools analyze tags on GitHub and provide visualizations [12], identify issue characteristics [16], support in understanding the network of programming languages used by users [5], and so on.

Visualizing certain aspects of a project or repositories provide better comprehension about the project to developers [20]. Developers are facilitated with visualization of source code architecture [11], the evolution of the project [21] and so on. However, such visualizations are generally complicated and do not explicitly provide pointers about source code or maintenance. Providing such information might be of interest to novice developers who wish to contribute to repositories on GitHub [3].

The usage of badges on crowd-sourced platforms had a positive impact on new users of the platform [24]. Some repositories specific to Nodejs use *npm-badges*[3] to represent a repository's characteristics, which are related to deployment and build details and hence provide a limited view of a repository specifically for the package called npm. Such badges are rarely valuable for novice developers who want to contribute to the repository based on the state of project maintenance, which could be useful for their onboarding.

---

[1]https://github.com/search?q=type:user&type=Users
[2]https://github.com/search?q=is:public
[3]https://www.npmjs.com/package/badges

Legay et al. [15] conducted a study on badges corresponding to build status, code coverage and so on, adopted by PHP and Rust package registries and observed that these badges could help developers in contributing to projects by instantly providing them details about a particular project across certain measures. Moreover, contributors in open-source projects volunteer their efforts and generally dedicate small amount of time to contribute and in the event of selecting poorly maintained projects, developer collaboration gets harder and such projects are likely to fail[7, 23].

Therefore, using badges as visual cues could help GitHub users to identify projects of their choice, based on the maintenance and source code quality of the project. Developer onboarding on GitHub has been studied in the literature to understand the factors that influence developer onboarding. Existing social relationships and prior language proficiency and experience were observed to be some of these factors [4]. NNLRank, a neural network model to suggest projects for better developer onboarding, based on developer skills and project features, has also been proposed [18]. This network is based on a ground-truth list of projects and considers only the developers' social profile and commit history of the projects. However, it does not consider the maintenance aspects of the project.

Hence, we propose *GitQ*, as a tool towards adding badges such as `Active Authors Rate 1.59%`, `Portion Impacted (LOC) 0.20%` as visual cues to GitHub repositories. Such visual cues could help developers in understanding source code and maintenance metrics, without exploring repositories in detail. *GitQ* displays metrics related to source code of the repository and maintenance status of the repository that could help in understanding the activeness and complexity of repositories, which are not explicitly present in the existing literature. We further evaluated *GitQ* through a user survey, with 15 volunteers, using a 5-point Likert scale-based questionnaire adapted from the UTAUT technology acceptance model. Around 73% of the participants found *GitQ* to be useful and to be providing a good user experience. Evaluation results and questionnaire can be found here[4].

2 RELATED WORK

GitHub provides various functionalities that help developers in developing, maintaining and extending their software. For instance, stargazers count, pull request lead time, programming languages used, act as popularity indicators and could be of interest to the contributors [9]. However, researchers have identified certain drawbacks on the existing GitHub platform such as difficulties in understanding commit messages, merged pull requests not being visible, unclear repository information, and so on [13], which can hinder the developer's productivity and cause a negative impact on user experience. Visual cues such as badges have been implemented in crowd-sourced software platforms such as Stack Overflow, StackExchange [24] and GitHub to enhance user experience [25].

Visual Cues On Stack Overflow - Stack Overflow[5] provides badges to the questions based on the programming language used, the context of the problem, and so on. Venigalla et al. have proposed to tag questions on Stack Overflow based on their context and

assign corresponding contextual badges [26]. Beyer et al. have also attempted to add badges on Stack Overflow based on the purpose of the questions on StackOverflow, which was further observed to be useful to Stack Overflow users [2].

Visual Cues On GitHub - A study on npm-badges assigned by project maintainers to repositories on GitHub revealed that badges such as build status, up-to-dateness, test coverage, and so on were primarily identified to be useful to contributors and users on GitHub [25]. *npm-badges* on GitHub repositories have facilitated researchers to analyze and obtain insights on types of continuous integration services, dependency managers adopted and underlying software tools used, across multiple repositories on GitHub [14]. A study by Prana et al. revealed that most software professionals found the presence of automated labeling of GitHub content, in the form of badges, to be useful and to facilitate easy discovery of information [22]. Legay et al. have observed that most of the information presented by badges is static, highlighting the build status of the projects [15].

The npm-badges are to be manually added by the repository owners and are static in nature. However, the deployment and build details provided through npm-badges do not provide maintenance related information such as the developer community in the repository, issue resolution rate and so on, which could be useful for developer onboarding.

Though badges or labels, automatically assigned, on crowd-sourced platforms such as StackExchange and Stack Overflow were observed to impact novice developers positively, usage of such automated visual cues is not yet explored on GitHub [24, 26]. Automatically adding badges as visual cues to GitHub repositories could help novice developers gain better insights about the repositories, thus improving user experience on GitHub [22]. However, to the best of our knowledge, no tools or approaches have been presented to add automated badges based on source code and maintenance characteristics of GitHub repositories, other than for categorizing readme-files [22].

Hence, we present *GitQ*, a Google Chrome plugin, aimed to automatically annotate GitHub with badges based on metrics for source code `Packages 15568` and maintenance `File Dependencies 8115`, generalized for a majority of GitHub repositories.

3 DESIGN AND DEVELOPMENT OF *GITQ*

We designed *GitQ* as a light-weight browser extension to convey essential information about a project. *GitQ* quantifies the project's architecture and maintenance with few metrics as visual cue badges for the GitHub repositories, thereby assisting developers to gain insights. Metrics provide information about the repository that is not explicitly mentioned in the existing documentation, such as readme files, source code comments, and so on. *GitQ* displays two categories of metrics - 1) source code metrics and 2) project maintenance metrics.

We have defined the source code metrics to include dependencies among files in the repository, number of inherited classes, number of overridden methods, and so on in the repository across all files with the ".java" extension. These metrics are based on the commonly used CK Metrics [6] defined to identify code quality for object-oriented languages. Considering the hierarchical organization of
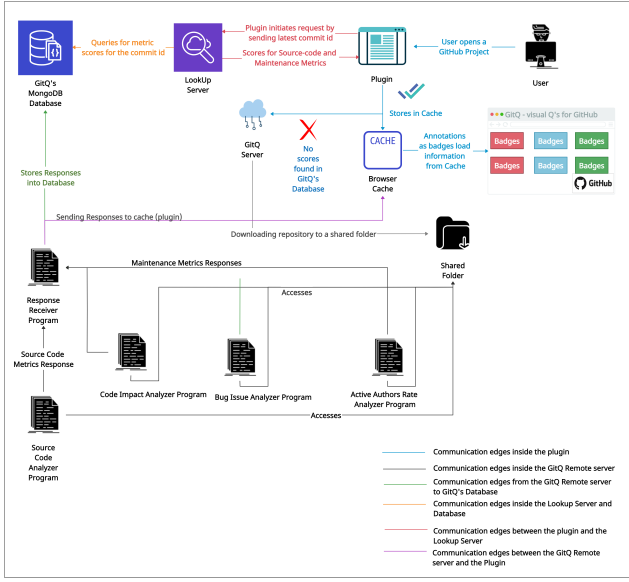
Figure 1: Workflow diagram of *GitQ*

repositories, CK metrics defined for object-oriented languages could align with the structure and hierarchy of GitHub repositories. The maintenance metrics include active authors rate, ratio of open and closed bug issues and extent of impact of commits on files and lines of code. These metrics could help in understanding the activeness of repository contributors, issue resolution rate in the repository and also average impact of commits in the repository. Detailed information about the insights shown by icons and definition of metrics are presented here[6]. Thus, *GitQ* intends to portray some of the project's complex information as a visual feed to help developers better investigate the projects to contribute.

We implemented *GitQ* as a browser extension, which displays badges on the GitHub repository page by sending the Project URL to a flask [7] server. Figure 1 depicts the work-flow of *GitQ*. The back-end downloads the repository and then calculates values for the metrics proposed. It then sends the values back to the plugin, resting on the client's page. The starter program in the server receives a download request from the plugin. Metric calculations are routed at different URLs; hence metric calculations are executed in parallel. Python functions mapped to the routes access the shared folder (contains the downloaded repository) for processing metrics. Every response from the route is then sent back to the plugin as a JSON object. The repository is then automatically deleted. On the plugin side, the metric scores are stored inside the browser cache by mapping scores to commit id. Therefore, a project once analyzed, can be displayed with badges within no time for future visits until a new commit has been made to the repository.

4 USER SCENARIO

Consider *Moksha*, a novice developer, willing to contribute to projects on GitHub to enhance her project development skills. She intends

---

to select projects that are well maintained, but confused about choosing the right project to contribute. She wishes to channel projects that are fairly maintained and well coded (less buggy) to get a glimpse of the project hierarchy and its maintenance over time. She discovers *GitQ* as a tool that might reduce her effort in identifying the right repository. To use *GitQ*, she loads the plugin into the browser and opens *ReactiveX/RxJava*[8] (as depicted in Figure 2[A]). *GitQ* produces the report for the entire repository, as shown in Figure 2[B]. She finds additional information about the project at a glance in the form of badges such as `Inherited Classes 2943`, `Bug Issues (Open:Closed) 0:262`. For example, Moksha hovers onto Bugs icon (left-most icon in the Figure 2[C]) and finds that the project has excellent bug handling but poor code composition. She discovers that at least 50% of the code is affected for every commit. Moreover, she also observes that the project has a fairly inactive community (Less than 10% of the authors remain active at any time); this would help her understand the activeness of a contributor in the project. Thus with *GitQ*, Moksha gains better insights about the repository. This could help her escape the tedious process of manually browsing through the source code of the repository and its history.

5 EVALUATION AND RESULTS

GitQ was developed to improve developers' experience while using GitHub and help them identify the right repository for reuse or contribution. Hence, we decided to evaluate *GitQ* with respect to developers' perception towards its usability, usefulness, and correctness. The UTAUT (Unified Theory of Acceptance and Use of Technology) [27] model comprises of Performance Expectancy (PE), Effort Expectancy (EE), Social Influence (SI), Facilitating Conditions (FC) and Behavioural Intention to Use (BI) dimensions. Considering the need for *GitQ* to be evaluated against similar dimensions, we chose to evaluate *GitQ* using UTAUT model.

We have added an additional dimension, Trust to fit the functionality of *GitQ*. We conduct a pilot user survey[15, 19] with 15 participants, to obtain insights on the user interpretation and usefulness of badge indication on a repository. We envision that obtaining insights on these aspects could be of potential use for future versions of *GitQ*. The 15 participant user survey comprised two research scholars, three undergraduate students, and ten professional developers working in the industry, with profound experience in using GitHub as a platform for open-source coding. A document explaining the steps involved in installing and using the plugin is circulated among all the volunteers along with the survey form. The participants were requested to view at least 20 projects on GitHub with *GitQ* plugin installed, to decide on the projects they wish to contribute, and to evaluate the correctness of the metrics used in *GitQ*. The survey form comprised 12 questions based on the UTAUT questionnaire and five questions aimed towards understanding the volunteers' demographics. The survey participants were asked to fill in the Likert-scale based questionnaire based on their experiences after using *GitQ*.

On an average the participants took around 10 minutes to view a repository and evaluate the correctness of metric values generated by *GitQ*. The results of the questionnaire presented in Figure 3 indicate that majority of the volunteers agreed that *GitQ* could be
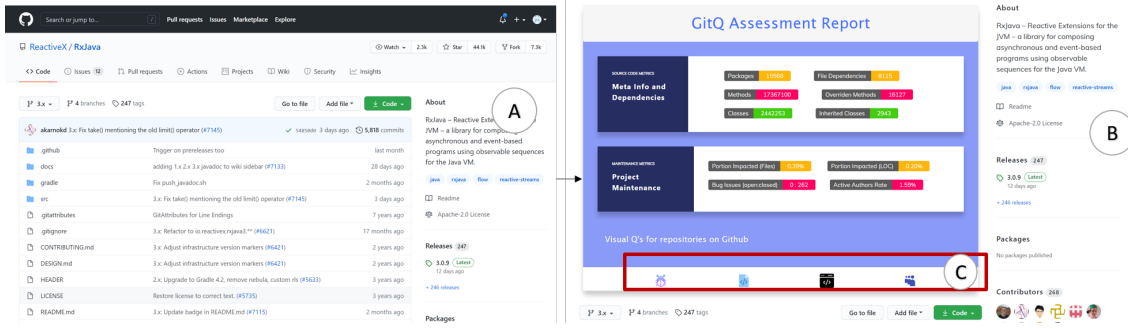

Figure 2: User Scenario, with [A] depicting GitHub page prior to badges being added, [B] indicating the badges augmented and [C] indicating the list of insight-icons for maintenance metrics.
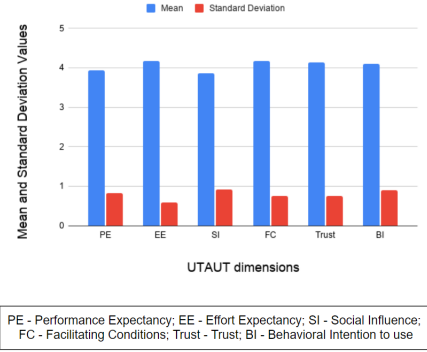

Figure 3: Results of *GitQ* User Survey

utilized as a indicator of project quality with the badges provided and might influence their decision in deciding GitHub projects (Mean > 3.5 and Standard Deviation(SD) < 1). The mean values of the PE, EE, SI and FC are close to 4, indicating that *GitQ* is easy to use with the existing technical expertise of the participants, and that *GitQ* could help in obtaining insights about the project. It has also been observed that participants widely relied on the metrics generated by *GitQ* (Mean = 4.13 and SD = 0.74), indicating a positive impact of the trust dimension. The users have pointed out that projects with many files take a longer time to load and have also suggested extending *GitQ* to display source code metrics for projects in other programming languages, apart from Java.

6 DISCUSSION AND LIMITATIONS

Currently *GitQ* is engineered in a way that the metric computation is handled solely by a remote server hosted at https://services.iittp.ac.in. Hiding the remote server, only the light-weight plugin is provided to the end user. However, the remote server might be inaccessible during downtime. Hence, we have decided to disentangle the server from the plugin and provide users with an alternative to use *GitQ*. With this backup version of *GitQ*[9], users can run the metric computation on their own machines by following simple instructions outlined in the readme[10]. While the alternative version

makes usability slightly difficult, it provides users with uninterrupted experience.

Currently, the scope for evaluating source code metrics is limited to java based projects, while maintenance metrics can be evaluated for any project in general. *GitQ* could be extended to evaluate source code metrics for multiple programming languages. Also, the survey with 15 respondents might indicate biased results and might not generalize the opinion of a larger section of users. Thus, we plan to perform a robust survey, using an improved questionnaire, with a larger number of GitHub developers and contributors.

7 CONCLUSION AND FUTURE WORK

Developers coordinate their efforts into building projects on GitHub. To enhance the efficiency and user experience of developers, researchers have proposed various additions to the GitHub platform. The automatic addition of badges to content on open source platforms such as Stack Overflow improves the developer's understanding of its content. Hence, we proposed *GitQ*, as a Chrome plugin, to augment GitHub with badges that deliver visual cues quantifying the project's architecture and maintenance.

The plugin forms the front-end of *GitQ*, while the flask server in the back-end calculates values for the metrics chosen. Metrics and their values are shown as badges (Inherited Classes 2943 , Packages 15568) on the screen. Source code metrics were identified to be in coherence with CK metrics. Apart from the badges, a summary of the project's maintenance is shown at the bottom of the assessment report for each of the maintenance metrics. From the developer survey, we found that majority of the volunteers agree (mean of PE=3.94) with *GitQ* as a quick insight assistant to increase productivity and that they might adopt *GitQ* while using GitHub for their project development . They also agree to recommend *GitQ* to their peers.

As our goal is to propose the idea of using badges as quality indicator for GitHub projects, we demonstrated use of badges using a few metrics. The metrics currently present in *GitQ* are an effort to signal underlying project quality and maintenance. However, there is a tremendous scope to include metrics that better qualify for the definition of a well maintained or a good quality project. Hence, as a part of the future work, we intend to include more metrics that can identify the underlying best-practices of open-source projects. We also intend to display trends (in the form of graphs) observed in open-source projects to motivate developers to follow best practices.

---

[9]https://github.com/gitq-for-github/plugin/tree/master/%5Bstandalone-server-setup%5Dbackup-version
[10]https://github.com/gitq-for-github/plugin/blob/master/%5Bstandalone-server-setup%5Dbackup-version/README.md

REFERENCES

[1] Andrew Begel, Jan Bosch, and Margaret-Anne Storey. 2013. Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder. *IEEE Software* 30, 1 (2013), 52–66.

[2] Stefanie Beyer, Christian Macho, Massimiliano Di Penta, and Martin Pinzger. 2018. Automatically classifying posts into question categories on stack overflow. In *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*. IEEE, 211–21110.

[3] Hudson Borges and Marco Tulio Valente. 2018. What's in a GitHub star? understanding repository starring practices in a social coding platform. *Journal of Systems and Software* 146 (2018), 112–129.

[4] Casey Casalnuovo, Bogdan Vasilescu, Premkumar Devanbu, and Vladimir Filkov. 2015. Developer onboarding in GitHub: the role of prior social links and language experience. In *Proceedings of the 2015 10th joint meeting on foundations of software engineering*. 817–828.

[5] Dorota Celińska and Eryk Kopczyński. 2017. Programming languages in github: a visualization in hyperbolic plane. In *Eleventh International AAAI Conference on Web and Social Media*.

[6] Shyam R Chidamber and Chris F Kemerer. 1994. A metrics suite for object oriented design. *IEEE Transactions on software engineering* 20, 6 (1994), 476–493.

[7] Jailton Coelho and Marco Tulio Valente. 2017. Why modern open source projects fail. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. 186–196.

[8] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on computer supported cooperative work*. ACM, 1277–1286.

[9] Felipe Fronchetti, Igor Wiese, Gustavo Pinto, and Igor Steinmacher. 2019. What attracts newcomers to onboard on oss projects? tl; dr: Popularity. In *IFIP International Conference on Open Source Systems*. Springer, 91–103.

[10] Mohammad Gharehyazie, Baishakhi Ray, and Vladimir Filkov. 2017. Some from here, some from there: Cross-project code reuse in github. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 291–301.

[11] Gharib Gharibi, Rashmi Tripathi, and Yugyung Lee. 2018. Code2graph: automatic generation of static call graphs for python source code. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 880–883.

[12] Javier Luis Cánovas Izquierdo, Valerio Cosentino, Belén Rolandi, Alexandre Bergel, and Jordi Cabot. 2015. Gila: Github label analyzer. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, 479–483.

[13] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. 2014. The promises and perils of mining GitHub. In *Proceedings of the 11th working conference on mining software repositories*. ACM, 92–101.

[14] Hemank Lamba, Asher Trockman, Daniel Armanios, Christian Kästner, Heather Miller, and Bogdan Vasilescu. 2020. Heard it through the Gitvine: an empirical study of tool diffusion across the npm ecosystem. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 505–517.

[15] Damien Legay, Alexandre Decan, and Tom Mens. 2019. On the Usage of Badges in Open Source Packages on GitHub.. In *BENEVOL*.

[16] Zhifang Liao, Dayu He, Zhijie Chen, Xiaoping Fan, Yan Zhang, and Shengzong Liu. 2018. Exploring the characteristics of issue-related behaviors in GitHub using visualization techniques. *IEEE Access* 6 (2018), 24003–24015.

[17] Antonio Lima, Luca Rossi, and Mirco Musolesi. 2014. Coding together at scale: GitHub as a collaborative social network. In *Eighth International AAAI Conference on Weblogs and Social Media*.

[18] Chao Liu, Dan Yang, Xiaohong Zhang, Baishakhi Ray, and Md Masudur Rahman. 2018. Recommending github projects for developer onboarding. *IEEE Access* 6 (2018), 52082–52094.

[19] Paul W McBurney and Collin McMillan. 2014. Automatic documentation generation via source code summarization of method context. In *Proceedings of the 22nd International Conference on Program Comprehension*. 279–290.

[20] Leonel Merino, Mohammad Ghafari, and Oscar Nierstrasz. 2018. Towards actionable visualization for software developers. *Journal of software: evolution and process* 30, 2 (2018), e1923.

[21] Renato Lima Novais, André Torres, Thiago Souto Mendes, Manoel Mendonça, and Nico Zazworka. 2013. Software evolution visualization: A systematic mapping study. *Information and Software Technology* 55, 11 (2013), 1860–1883.

[22] Gede Artha Azriadi Prana, Christoph Treude, Ferdian Thung, Thushari Atapattu, and David Lo. 2019. Categorizing the content of GitHub README files. *Empirical Software Engineering* 24, 3 (2019), 1296–1327.

[23] Dirk Riehle, Philipp Riemer, Carsten Kolassa, and Michael Schmidt. 2014. Paid vs. volunteer work in open source. In *2014 47th Hawaii International Conference on System Sciences*. IEEE, 3286–3295.

[24] Tiago Santos, Keith Burghardt, Kristina Lerman, and Denis Helic. 2020. Can Badges Foster a More Welcoming Culture on Q&A Boards?. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 14. 969–973.

[25] 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*. 511–522.

[26] Akhila Sri Manasa Venigalla, Chaitanya S Lakkundi, and Sridhar Chimalakonda. 2019. SOTagger-Towards Classifying Stack Overflow Posts through Contextual Tagging (S).. In *SEKE*. 493–639.

[27] Viswanath Venkatesh, Michael G Morris, Gordon B Davis, and Fred D Davis. 2003. User acceptance of information technology: Toward a unified view. *MIS quarterly* (2003), 425–478.

[28] Yun Zhang, David Lo, Pavneet Singh Kochhar, Xin Xia, Quanlai Li, and Jianling Sun. 2017. Detecting similar repositories on GitHub. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 13–23.