

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/331718820>

Should I Stale or Should I Close? An Analysis of a Bot That Closes Abandoned Issues and Pull Requests

Conference Paper · March 2019

DOI: 10.1109/BotSE.2019.00018

CITATIONS

8

READS

395

4 authors:



Mairieli Wessel

University of São Paulo

15 PUBLICATIONS 84 CITATIONS

[SEE PROFILE](#)



Igor Steinmacher

Federal University of Technology - Paraná/Brazil (UTFPR)

126 PUBLICATIONS 1,412 CITATIONS

[SEE PROFILE](#)



Igor Scaliante Wiese

Federal University of Technology - Paraná/Brazil (UTFPR)

73 PUBLICATIONS 457 CITATIONS

[SEE PROFILE](#)



Marco Aurelio Gerosa

Northern Arizona University

219 PUBLICATIONS 2,954 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



AulaNet [View project](#)



Arquigrafia [View project](#)

Should I Stale or Should I Close? An Analysis of a Bot that Closes Abandoned Issues and Pull Requests

Mairieli Wessel
University of Sao Paulo
Sao Paulo, SP, Brazil
mairieli@ime.usp.br

Igor Steinmacher
Northern Arizona University
Flagstaff, AZ, USA
igor.steinmacher@nau.edu

Igor Wiese
Fed. Univ. of Technology, Parana
Campo Mourao, PR, Brazil
igor@utfpr.edu.br

Marco A. Gerosa
Northern Arizona University
Flagstaff, AZ, USA
marco.gerosa@nau.edu

Abstract—On GitHub, projects use bots to automate predefined and repetitive tasks related to issues and pull requests. Our research investigates the adoption of the *stale bot*, which helps maintainers triaging abandoned issues and pull requests. We analyzed the bots’ configuration settings and their modifications over time. These settings define the time for tagging issues and pull request as stale and closing them. We collected data from 765 OSS projects hosted on GitHub. Our results indicate that most of the studied projects made no more than three modifications in the configurations file, issues tagged as bug reports are exempt from being considered stale, while the same occurs with pull requests that need some input to be processed.

Index Terms—bots, open source software, abandoned issues

I. INTRODUCTION

Bots are software agents that integrate their work with humans’ tasks [1], serving as conduits between users and services [2] and performing complex tasks that cannot be entirely automated [3]. Bots support technical and social software engineering activities [4], such as communication and decision-making [2].

Open Source Software (OSS) projects hosted on GitHub have been adopting bots to automate a variety of predefined tasks, such as ensuring license agreement signing, reporting continuous integration failures, triaging issues, reviewing code and pull requests, and assigning reviewers [5]. Some studies investigate the impact of bots recommending reviewers [6] and helping developers to detect early duplicate development effort [7].

Triaging stale or dormant issues and pull requests is also supported by bots on GitHub. For example, GitHub created the *stale bot*.¹ This bot is described as a GitHub App that help maintainers by automatically labeling and closing abandoned issues and pull requests. Any project hosted on GitHub may leverage the features provided by this bot, and change its settings to adapt the characteristics for each project.

Maintainers leverage the *stale bot* to help cleaning and keeping the issue tracker and pull requests up-to-date, once one of the common reasons to close a pull request is when it is no longer relevant, as the project has progressed [8]. Moreover, the lack of feedback in pull-requests may discourage further contributions [9]; while issues left open for a long time may become outdated and, ultimately, confuse newcomers [10].

In this paper, we investigated OSS projects that adopted the *stale bot*. Specifically, we looked into the bots’ configuration settings defined by each project, and we analyzed how these projects adapt and maintain the bot over time. We collected data from 765 OSS projects that adopted the *stale bot*. We found that issues tagged as a bug report and pull requests that

need input from the developer, review or something else to be processed, are exempt from being considered abandoned or stale. Our results also indicate that in approximately 83% of the projects the configuration file was modified three times or less.

II. MOTIVATION SCENARIO

Ada, who works in an open source project that receives many daily contributions, spends considerable time to maintain/triage the issues and review contributions. She wants to be more proactive by focusing on issues and pull requests that are affecting the projects and the developers, but the amount of open pull requests make it hard to manage. A developer suggested to try out *stale bot* and explained how it works:

- After a period of inactivity predefined by maintainers, a label is added to mark an issue/pull request as stale.
- When an issue/pull request is marked as stale, optionally a comment is posted to notify the stakeholders.
- The issue/pull request is automatically closed if no more activity occurs. If the issue/pull request is updated, then the stale label is removed.

Considering this suggestion, Ada becomes interested in adopting *stale bot* but worries that it might seem hostile or offensive for contributors, especially for newcomers. She figures out that some bot settings can be modified to adapt the stale characteristics to the project, including changing the time interval to consider issues and pull requests stale, and the messages that the bot send. She believes that, by adjusting the settings, it is possible to decrease the chances of labeling and closing issues and pull requests that had recent activity, and that this can be used to make maintainers aware of issues and pull requests that need their attention, what may help newcomers.

III. METHOD

This study aims to explore how OSS projects on GitHub use the *stale bot*, answering the following research questions:

RQ1. What are the characteristics of stale issues and pull requests?

We aimed to investigate the configuration settings of *stale bot* as defined by maintainers to understand which characteristics commonly define issues and pull requests to be considered stale in OSS projects.

RQ2. How stable is the bot configuration for a project?

¹<https://probot.github.io/apps/stale/>

In this research question, we aim to understand how much effort is required to use and maintain the bot on issues and pull requests of OSS projects hosted on GitHub. To do it, we analyzed the number and frequency of changes in the bot configuration file.

A. Project Selection

We selected OSS projects hosted on GitHub that adopted the *stale bot*. To identify these projects, we verified whether they had the required configuration file (*.github/stale.yml*). Using GitHub’s public dataset on Google BigQuery,² we used a query to search for public projects that contained this configuration file. We started with 1,484 projects. To further refine our dataset, we decided to exclude forks and deleted projects. As a result, we obtained 770 projects. For each project, we used the PyDriller [11] to collect the data of commits that modified the *.github/stale.yml* file.

As mentioned before, a *.github/stale.yml* file is required in the default branch to enable the bot on the repository. In this file, the project maintainers may change some default settings:

- **daysUntilStale** (defaults to 60 days). Number of days of inactivity before an issue or pull request becomes stale.
- **daysUntilClose** (defaults to 7 days). Number of days of inactivity before an issue or pull request with the stale label is closed. If disabled, issues still need to be manually closed.
- **staleLabel** (defaults to “wontfix”). Label to use when marking the issue or pull request as stale. A project can have only one stale label.
- **exemptLabels** (defaults to “pinned” and “security”). Issues or pull requests with these labels are never considered stale.
- **only** (defaults to “issues” and “pulls”). Restricts bot action to “issues” or “pulls” only. If empty, the bot work on both “issues” and “pulls”.

Besides the settings presented above, it is possible to specify configuration settings that are specific to just “issues” or “pulls.”

To extract these settings from the source code of configuration file, we generated a parser in Python. Thus, we discarded five selected projects that did not follow the YAML format on the *.github/stale.yml* file. At the end of this process, we obtained 765 projects.

B. Data analysis

We analyzed bot configuration settings of 765 software projects, investigating the days until stale, days until close, stale label, and exempt labels to find which characteristics commonly define issues and pull requests that are considered stale in OSS projects (RQ1).

As previously explained in Section III-A, it is allowed override the configuration settings specific to “issues,” “pull requests,” or both (we called it “all”). We conduct all analyzes by distinguishing these categories of settings, examining both frequency and type of modifications made in the configuration file (RQ2).

For more details about the projects used and replication purposes, we made our data and source code publicly available.³

²https://bigquery.cloud.google.com/dataset/bigquery-public-data:github_repos

³<https://github.com/mairieli/botSE-2019>

IV. RESULTS

In the following, we present the results according to the research questions.

A. RQ1. What are the characteristics of stale issues and pull requests?

To characterize stale issues and pull requests, we investigated the *stale bot* configuration file for each selected project. The first analysis we conducted aimed to understand whether the bot was being used to identify stale issues or pull requests. We found 75 projects ($\approx 9.8\%$) that adopted the bot only for issues, 19 ($\approx 2.5\%$) that adopted only for pull requests, and 671 ($\approx 87.7\%$) that adopted the *stale bot* for both issues and pull requests. By investigating these 671 projects, we noticed that most of them maintained a common setting for issues and pull requests, and 12.2% (82 projects) specified the different settings for “issues” and “pull requests.”

After analyzing the *only* setting and specific configurations, we looked into the settings that define the number of days until the bot considers an issue or pull request stale. Figure 1 presents the distribution of *daysUntilStale* setting for our sample. Our data show that this distribution is different according to the target set for the bot (issues, pull requests, or both (all)). For the issues settings $Q1 = 30$, median = 30, $Q3 = 84$; for the pull requests settings $Q1 = 14$, median = 14, $Q3 = 30$, and for both $Q1 = 30$, median = 60, $Q3 = 60$. Pull requests have been set to become stale in fewer days than issues. We also noticed some extreme cases; for example, we found eight projects that defined the *daysUntilStale* as more than 36,000 days. Therefore, we manually investigated each of these projects and noted they misunderstood the workflow of the bot.

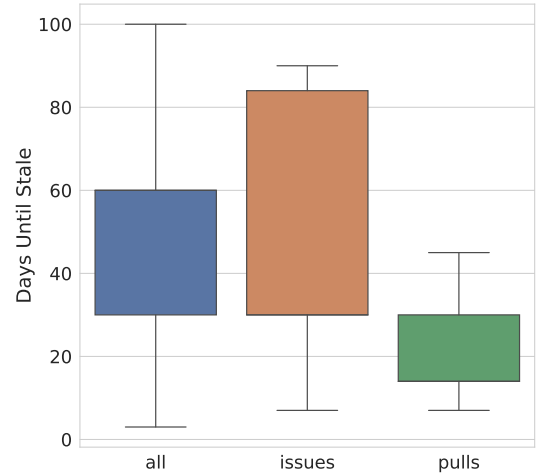


Fig. 1. Distribution of days until stale

As we did for *daysUntilStale* setting, we looked into the settings that define the number of days until the bot closes an issue or pull request marked as stale. Among the analyzed projects, 77 had this value set to *false* at the moment of our collection (disabling the feature that makes the bot automatically close the issues and pull requests). Figure 2 presents the distribution of *daysUntilClose* setting for the 688 ($\approx 90\%$) projects that kept the closing feature active. For the issues settings $Q1=7$, median=7, $Q3=40$; for the pull

requests settings Q1=7, median=7, Q3=14, and for all settings Q1=7, median=7, Q3=7. By looking at Figure 2, we noticed that the configurations that included both issues and pull requests (all), kept `daysUntilClose` with the default value (excluding outliers, not displayed in the figure).

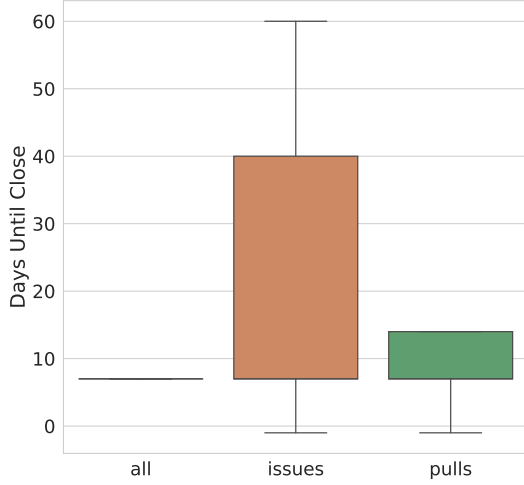


Fig. 2. Distribution of days until close

Analyzing the `staleLabel` settings, we found 16 different labels applied by the projects. By analyzing them, we found that four of them corresponded to more than 90% of all the labels adopted. Table I presents the frequency of these 4 stale labels. Interestingly, the default label “wontfix” is not the most commonly used. We hypothesize that projects changed it since “wontfix” is offered as a default label when one creates a new repository in GitHub. Thus, projects usually make use of this label and do not want to have it used with more than one meaning. Therefore, the project maintainers adopt other labels, preferring using “stale,” as it is possible to observe in the table.

TABLE I
FREQUENCY OF MOST COMMON STALE LABELS

Stale Label	Freq. in Issues (%)	Freq. in Pulls (%)	Freq. in All (%)
stale	125 (79.6%)	26 (25.7%)	303 (51.4%)
wontfix	25 (15.9%)	10 (9.9%)	245 (41.5%)
abandoned	-	60 (59.4%)	2 (0.3%)
inactive	3 (1.9%)	3 (2.9%)	12 (2%)

* Gray rows represent the default settings.

We also analyzed the labels assigned to issues and pull requests that maintainers never want from being considered stale. To do so, we manually inspected the labels defined under the `exemptLabels` setting. Table II shows the most recurrent exempt types of labels used (it is worth mentioning that we further classified the labels in more generic categories).

Regarding the issues’ settings, we noticed that the distribution of labels used for issues and pull requests are different. The most frequently exempt label category for issues represents bug reports (usually labeled as “bug”), appearing in 67.6% of the projects. Enhancements closely follow this label. Apart from the most common ones, one interesting finding is that a non-negligible number of projects set the

issues identifying newcomer-friendly tasks as exempt from staling. These issues are usually low-hanging fruits and may benefit newcomers’ onboarding, so they are not resolved by more active members.

For the pull requests’ settings, the most common label categories represent the contributions that need input from the developer (“waiting for CLA”), review (“needs review”), or something else (“on hold”) to be processed.

TABLE II
FREQUENCY OF MOST COMMON EXEMPT LABELS

Exempt labels	Freq. in Issues (%)	Freq. in Pulls (%)	Freq. in All (%)
security	61 (41.4%)	27 (26.7%)	360 (61.1%)
pinned	43 (27.3%)	26 (25.7%)	371 (62.9%)
bug	106 (67.6%)	5 (4.9%)	104 (17.6%)
enhancement	95 (60.5%)	3 (2.9%)	73 (12.3%)
on hold	60 (38.2%)	59 (58.4%)	-
newcomer	17 (10.8%)	6 (5.9%)	72 (12.3%)
maybe later	12 (7.6%)	8 (7.9%)	65 (11%)
documentation	69 (43.9%)	-	-
needs review	-	59 (58.4%)	20 (3.3%)
accepted	3 (1.9%)	-	60 (10.1%)
proposal	61 (38.8%)	1 (0.9%)	1 (0.1%)
waiting for CLA pass	-	59 (58.4%)	-
tests	59 (37.5%)	-	-
feature	7 (4.4%)	1 (0.9%)	45 (7.6%)
work in progress	4 (2.5%)	1 (0.9%)	36 (6.1%)
discussion	7 (4.4%)	-	20 (3.3%)
blocked	9 (5.7%)	1 (0.9%)	14 (2.3%)
under consideration	-	-	23 (3.9%)
no stale	5 (3.1%)	2 (1.9%)	13 (2.2%)

* Gray rows represent the default settings.

RQ1. Pull requests have been defined to become stale in fewer days than issues. Issues tagged as bug report are exempt from being considered stale, while the same occurs with pull requests that need some input to be processed.

B. RQ2. How stable is the bot configuration for a project?

After adding the first version of the configuration file to the project, maintainers may update the settings until reaching an appropriate configuration for the project. To understand this process, we analyzed the history of changes in the configuration file for all projects. We noticed that some projects take some time changing it until they reach the ideal configuration. Thus, we considered that the number of modifications is a proxy to the effort required to use the bot in a project.

From this analysis, we observed that 453 ($\simeq 59.2\%$) projects only added the configuration file and made no further modifications. Interestingly, 49 ($\simeq 6.4\%$) projects kept the configuration file exactly as provided in the bot page,⁴ maintaining the default settings.

In Figure 3, we present the distribution of the number of modifications in the `stale.yml` file per project. We considered only the 312 projects that changed the configuration file at least once after the addition, and we found that most projects (259, or $\simeq 83\%$) have modified the file between one and three times after its initial inclusion in the project.

Figure 3 also shows some outliers with 7, 8, 9, 16, and 59 modifications. The extreme case happened in the *SaltStack*

⁴<https://github.com/probot/stale#usage>

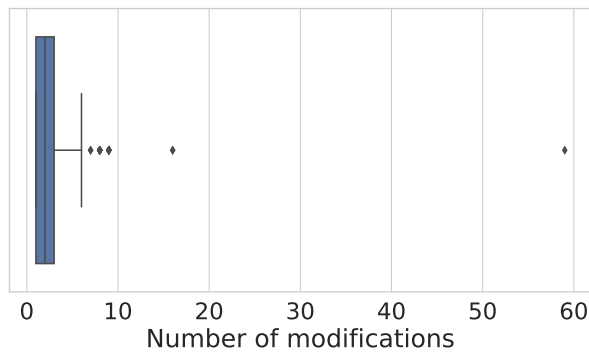


Fig. 3. Distribution of the number of modifications in *stale.yml* file

project.⁵ Figure 4 shows the timeline of modifications for this project. The configuration file was modified by 59 commits from May 12, 2017, to December 14, 2017. By analyzing the changes, we could observe that all of them were changes in the `daysUntilStale` setting, decreasing it by 30, 25, 15, and 10 days. All modifications followed the same structure in the commit message: “Reduce the number of days an issue is stale by X”. We could notice that this gradual reduction was to avoid overloading the project with a lot of stale issues and pull requests needed to be manually inspected in a short period. Until December 2018, more than 3,500 issues were closed by the bot in this project.

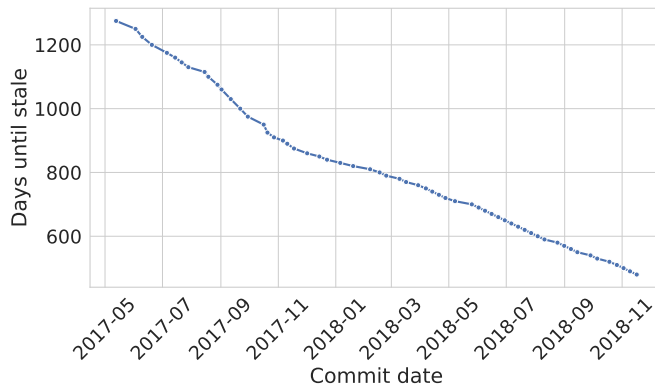


Fig. 4. The progress of days until stale

Looking into a pull request⁶ of the *atom* project, we found another evidence that projects have defined high values to the `daysUntilStale` setting and then decreased it. Manually analyzing another outlier, we found 17 commits with modifications to the *stale.yml* file in the *Moya*⁷ project. These modifications included addition of exempt labels and decreasing both `daysUntilStale` and `daysUntilClose` values. The goal of these changes were made clear by the committer: “... I think leaning on our bot is a good idea, and I can see us changing this to just a week sometime in the future.”

⁵<https://github.com/saltstack/salt>

⁶<https://github.com/atom/atom/pull/15492>

⁷<https://github.com/Moya/Moya>

RQ2. We observed that $\simeq 59.2\%$ of projects only added the file and considering the other projects, $\simeq 83\%$ of them made no more than three modifications. Therefore, the bot configuration settings are stable in most cases.

V. RELATED WORK

Bots are extensively proposed and analyzed in the literature of different domains, including social media [12]–[14], online learning [15]–[18], and Wikipedia [19], [20]. In Software Engineering, bots support several activities automating tasks that generally require human interaction [3]. While some bots support development and deployment, team and task management, and file sharing, others are used as social media, or even for fun [2], [5]. In this paper, we extend the existing knowledge on bots on GitHub by exploring a specific bot, analyzing how it is adopted and changed over time.

VI. CONCLUSION

In this paper, we analyzed how OSS projects hosted on GitHub use a bot that helps maintainers work by automatically labeling and closing abandoned or inactive issues and pull requests. Exploring the bot configuration settings (**RQ1**), we showed that most projects ($\simeq 87.7\%$) adopted the *stale bot* for both issues and pull requests. We also found that pull requests have been defined to become stale in fewer days than issues. We also found that abandoned issues tagged as bug report are generally exempt from staling. In addition, we investigated how often the bot configuration file changes (**RQ2**). We found that most projects made no more than three changes in the configurations file.

ACKNOWLEDGMENT

This work is supported by the CNPq (# 430642/2016-4) and FAPESP MCT/CGI.br (# 2015/24527-3).

REFERENCES

- [1] U. Farooq and J. Grudin, “Human-computer integration,” *interactions*, vol. 23, no. 6, pp. 26–32, Oct. 2016.
- [2] M.-A. Storey and A. Zagalsky, “Disrupting developer productivity one bot at a time,” in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2016. New York, NY, USA: ACM, 2016, pp. 928–931. [Online]. Available: <http://doi.acm.org/10.1145/2950290.2983989>
- [3] C. Lebeuf, M.-A. Storey, and A. Zagalsky, “Software bots,” *IEEE Software*, vol. 35, no. 1, pp. 18–23, 2018.
- [4] B. Lin, A. Zagalsky, M. Storey, and A. Serebrenik, “Why developers are slacking off: Understanding how software teams use slack,” in *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, ser. CSCW ’16 Companion. New York, NY, USA: ACM, 2016, pp. 333–336. [Online]. Available: <http://doi.acm.org/10.1145/2818052.2869117>
- [5] M. Wessel, B. M. de Souza, I. Steinmacher, I. S. Wiese, I. Polato, A. P. Chaves, and M. A. Gerosa, “The power of bots: Characterizing and understanding bots in oss projects,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, p. 182, 2018.
- [6] Z. Peng, J. Yoo, M. Xia, S. Kim, and X. Ma, “Exploring how software developers work with mention bot in github,” in *Proceedings of the Sixth International Symposium of Chinese CHI*, ser. ChineseCHI ’18. New York, NY, USA: ACM, 2018, pp. 152–155. [Online]. Available: <http://doi.acm.org/10.1145/3202667.3202694>
- [7] L. Ren, S. Zhou, C. Kästner, and A. Wasowski, “Identifying redundancies in fork-based development.”
- [8] G. Gousios, M. Pinzger, and A. v. Deursen, “An exploratory study of the pull-based software development model,” in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 345–355. [Online]. Available: <http://doi.acm.org/10.1145/2568225.2568260>

- [9] I. Steinmacher, G. Pinto, I. S. Wiese, and M. A. Gerosa, "Almost there: A study on quasi-contributors in open source software projects," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: ACM, 2018, pp. 256–266. [Online]. Available: <http://doi.acm.org/10.1145/3180155.3180208>
- [10] I. Steinmacher, I. S. Wiese, A. P. Chaves, and M. A. Gerosa, "Why do newcomers abandon open source software projects?" in *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE '13. IEEE, 2013, pp. 25–32.
- [11] D. Spadini, M. Aniche, and A. Bacchelli, *PyDriller: Python Framework for Mining Software Repositories*, 2018.
- [12] S. Savage, A. Monroy-Hernandez, and T. Höllerer, "Botivist: Calling volunteers to action using online bots," in *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. New York, NY, USA: ACM, 2016, pp. 813–822.
- [13] N. Abokhodair, D. Yoo, and D. W. McDonald, "Dissecting a social botnet: Growth, content and influence in twitter," in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, ser. CSCW '15. New York, NY, USA: ACM, 2015, pp. 839–851. [Online]. Available: <http://doi.acm.org/10.1145/2675133.2675208>
- [14] B. Xu, T. C.-W. Yuan, S. R. Fussell, and D. Cosley, "Sobot: Facilitating conversation using social media data and a social agent," in *Proceedings of the Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, ser. CSCW Companion '14. New York, NY, USA: ACM, 2014, pp. 41–44. [Online]. Available: <http://doi.acm.org/10.1145/2556420.2556789>
- [15] S. Ghose and J. J. Barua, "Toward the implementation of a topic specific dialogue based natural language chatbot as an undergraduate advisor," in *Informatics, Electronics & Vision (ICIEV), 2013 International Conference on*. Washington, DC, USA: IEEE, 2013, pp. 1–5.
- [16] A. M. Latham, K. A. Crockett, D. A. McLean, B. Edmonds, and K. O'Shea, "Oscar: An intelligent conversational agent tutor to estimate learning styles," in *International Conference on Fuzzy Systems*. Washington, DC, USA: IEEE, 2010, pp. 1–8.
- [17] K. Nakamura, K. Kakusho, T. Shoji, and M. Minoh, "Investigation of a method to estimate learners interest level for agent-based conversational e-learning," in *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Berlin, Heidelberg: Springer, 2012, pp. 425–433.
- [18] C. Roda, A. Angehrn, T. Nabeth, and L. Razmerita, "Using conversational agents to support the adoption of knowledge sharing practices," *Interacting with Computers*, vol. 15, no. 1, pp. 57–89, 2003.
- [19] R. S. Geiger and A. Halfaker, "Operationalizing conflict and cooperation between automated software agents in wikipedia: A replication and expansion of 'even good bots fight'," *Proc. ACM Hum.-Comput. Interact.*, vol. 1, no. CSCW, pp. 49:1–49:33, Dec. 2017. [Online]. Available: <http://doi.acm.org/10.1145/3134684>
- [20] D. Cosley, D. Frankowski, L. Terveen, and J. Riedl, "Suggestbot: Using intelligent task routing to help people find work in wikipedia," in *Proceedings of the 12th International Conference on Intelligent User Interfaces*, ser. IUI '07. New York, NY, USA: ACM, 2007, pp. 32–41. [Online]. Available: <http://doi.acm.org/10.1145/1216295.1216309>