GitRDone: Using Nudge Theory and Positive Reinforcement to Improve Git Usage Practices

Brittany DePoi, Lanae Blethen Kawa, & Brynn Evans

Technology Empowerment & Advancement, Cigna

Introducing GitRDoneBot!

GitRDoneBot is a developer tool that attempts to translate Good Git Practices into team behavior via comments and emojis from a bot teammate.

- **Does** automate feedback of good git practices from published work
- **Does not** attempt to automate tasks away from engineers who use git
- **Improves** git usage and understanding
- **Over time** teams will implicitly strengthen practices rather than explicitly being told what to do

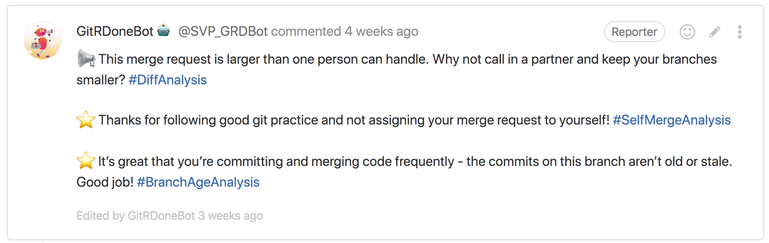


Figure 1. GitRDoneBot Comment Example

Problem & Motivation

GitHub and GitLab are powerful source code management and collaboration tools when leveraged correctly. Novice teams are not always aware of the advantages conferred by following some basic guidelines. Even the most experienced teams can form bad habits without the occasional reminder to use good practices. As teams develop consistently mature source control practices they achieve greater velocity and confidence throughout the software development lifecycle.

Acknowledged Good Git Practices for Pull/Merge Request Feedback

GitRDoneBot's parameters are derived from established good practices that have emerged through the consensus of the open source engineering community.

Merge/Pull Requests:

1. Contain a Manageable Number of Changes: *1 to 500 Lines of Diff*
   - The human brain can only process 500 lines of text at one time [8]
   - Most pull requests for open source projects are less than 20 lines long [4]
2. Contain Short-Lived Branches: *1 to 14 days*
   - The combination of trunk-based development with short-lived feature branches accelerates code development, especially for asynchronous and/or large teams [5]
   - Any branch that exists for more than 2 days runs the risk of becoming a long-lived branch.
3. Are Reviewed by Another Person: *Not self-assigned or self-merged*
   - The open source community benefits from many eyes on code, therefore code should not be introduced to the main working branch without being vetted by at least one other person [6]
4. Are Distributed Across the Team: *1 to 10 Requests assigned to a single person*
   - To remove bottlenecks, low quality reviews, increased context switching, and single-person dependencies, check to make sure that MR assignees don't have too many other MR reviews on their plate.
5. Do Not Contain Commented-Out Code (Beta)
   - This feature is an assumption that is pending further research before being fully implemented.
6. Contain Good Commit Messages (Beta):
   - Adopting established good practice commenting guidelines [2] can greatly improve the usefulness of built-in git tools like revert and log, and can make understanding the history of a project much easier.

Why GitRDoneBot?

1. **Novel application of automation** in the continuous integration/continuous deployment pipeline to improve engineering experience.
2. **Aggregates good practices** from published works.
3. **Supports team autonomy** through configuration of ranged thresholds and feedback allowing teams to determine what works for them.
4. **Aids accessibility and inclusion** by leveling the playing field so that novice technology users have equal access to the information they need to form good git habits.
5. **Caters to the emotional** aspects of interaction by providing positive reinforcement.

Automated Feedback That Matters

Using Nudge Theory

GitRDoneBot is inspired by the concept of nudge theory from the discipline of behavioral economics. A nudge is an action that increases the likelihood of an individual making a particular choice or behaving in a particular way. Nudges trigger automatic cognitive processes in favor of a desired outcome and have been proven to be one of the best tools to foster lasting behavioral changes [9]

How It Works

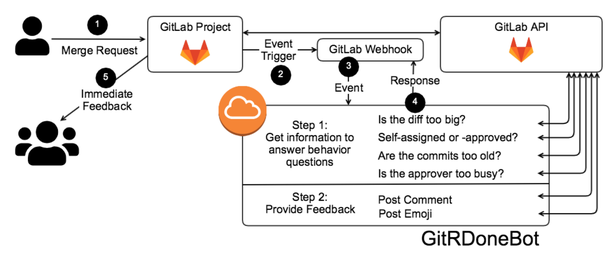


Figure 2. GitRDoneBot Flow Diagram

1. **Integrated**: Teams go about their normal activity opening merge requests.
2. **Simple Setup**: GitRDoneBot is added to GitLab projects once via a simple webhook trigger.
3. **Event Driven**: GitRDoneBot will receive the event when a merge request is created/updated/merged.
4. **Configurable Logic**: GitRDoneBot will analyze the event for a variety of behaviors, and will grab additional information via the GitLab API when needed.
5. **Real-time Feedback**: GitRDoneBot provides feedback derived from the results of its analysis in the form of comments and emojis that are pushed to the project in real time, every time.

Related Work

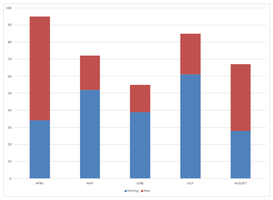
Before the design process began, a survey was performed of existing products that integrate with GitLab or GitHub. It was determined that none of the existing products met all of the goals.

- **GitPrime** [3] is a paid service that is geared towards gathering metrics, whereas the need being addressed required something that would provide feedback and education.
- **ProBot** [7] is an open source collection of apps focused on automating tasks, rather than providing feedback to help teams learn good practices.

Initial Results

The preliminary analysis includes the 204 projects that have added GitRDoneBot since March 2019.

Lines of Diff



(a) Active Projects Per Month

(b) Pct. Requests with > 500 Lines of Diff

Early numbers show two promising percentage decreases for large total diffs from March to May and again from June to August.

Between May and June there was a spike of extra large requests. A decrease in overall active users was also observed during that time, but interestingly, it is not the time period with the most new users. Analysis will continue to see if a trigger for this spike can be identified or if the data was skewed by outliers.

Self-Approved Requests

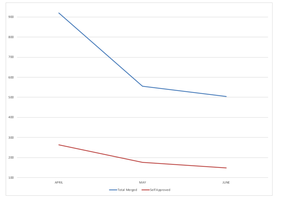


Figure 4. Self-Approved vs. Total Merged

Trends in self-approvals show a slight decrease in raw numbers over time.

However, when compared to the total number of merges the trend of self-approved merges hovers consistently around 30% for the three months of data currently available.

Analysis of all historical merge request GitLab data prior to June 2019 show that 31% of merges were self-approved, which is consistent with the findings above. This reveals a significant opportunity for code quality improvement by reducing this percentage. Work will continue to see if GitRDoneBot will be more effective over a longer period of time, or can be made more effective in general.

What's Next

Quantitative analysis of broad trends will continue, but focus will also shift to qualitative analysis and individual teams. There are many factors which influence git practices at the team level, so following a few teams over time in addition to monitoring overall trends will tell a more comprehensive story.

[1] Font Awesome.
User icons.
fontawesome.com/icons/user, 2019.

[2] Scott Chacon and Ben Straub.
Pro Git: Everything You Need to Know About Git.
Apress, 2nd edition, 2014.

[3] GitPrime.
Software engineering metrics for engineering leaders and management.
www.gitprime.com, 2019.

[4] G. Gousios, M. Pinzger, and A. van Deursen.
An exploratory study of the pull-based software development model.
www.gousios.gr/pub/exploration-pullreqs.pdf, 2014.

[5] Paul Hammant.
Trunk based development.
trunkbaseddevelopment.com/short-lived-feature-branches, 2018.

[6] Chris F. Kemerer and Mark C. Paulk.
The impact of design and code reviews on software quality: An empirical study based on psp data.
http://www.pitt.edu/ ckemerer/PSP_Data.pdf, 2009.

[7] ProBot.
Probot: Github apps to automate and improve your workflow.
probot.github.io, 2019.

[8] SmartBear Software.
Best practices for code review.
smartbear.com/learn/code-review/best-practices-for-peer-code-review, 2019.

[9] Richard H. Thaler and Cass R. Sunstein.
Nudge: Improving Decisions About Health, Wealth, and Happiness.
Penguin, 2009.