Product Vision
Octopeer for BitBucket

Out of Context

Date:
04-05-2016

Group members:
Arthur Guijt
4377338
a.guijt@student.tudelft.nl

Lars Stegman
4365801
l.s.stegman@student.tudelft.nl

Laurens Kroesen
4350286
l.kroesen@student.tudelft.nl

Cas Buijs
4345185
s.j.m.buijs@student.tudelft.nl

Thomas Overklift
4080890
t.a.r.overkliftvaupelklein@student.tudelft.nl

Responsible Teacher:
Alberto Bacchelli
a.bacchelli@tudelft.nl

Teaching Assistants:
Bastiaan Reijm
A.B.Reijm@student.tudelft.nl
Aaron Ang
A.W.Z.Ang@student.tudelft.nl

An introduction to ‘Octopeer for Bitbucket’

The Octopeer for BitBucket (Octopeer) plugin offers users an insight into their pull request review behaviour, but for whom is the Octopeer plugin useful? The Octopeer plugin tracks the behaviour of software engineers during their code review on pull requests and allows them to find patterns in the data. This is not only useful for the developers themselves, but allows software engineering researchers to gather data for their research. Besides the advantages for individual developers, the Octopeer extension allows software companies to have insight in their developers’ talents, which helps them use the developers more efficiently.

When a developer wants to have his or her code added to an existing code base, he creates a pull request. This pull request might be reviewed by other developers on the code base. This review might be done in different degrees of thoroughness. If a reviewer is very thorough it is less likely that bugs or other unexpected behaviour will enter the code base. If a reviewer reads through the request very quickly and doesn’t test any of the code, the chances of something bad happening increase. Enabling developers to monitor their behaviour, allows them to change it if necessary.

Octopeer for Researchers

Software engineering researchers want to gain deeper insight into how developers build software. Understanding how developers work together and individually, allows the researchers to develop new techniques, theories and teach future engineers the best practices. (Van Deursen, 2009)

However, to be able to research topics like pull request behaviour and other version control practices, researchers need data. The Octopeer chrome extension allows researchers to gather data on the behaviour of developers during their pull request reviews. The Octopeer extension tracks multiple factors like the amount of time that has passed between first looking at a pull request and approving or merging it. There have been multiple researches looking at the behaviour of developers during their pull requests. In the paper by Bacchelli et al. (2014b), they mention how they collected data by interviewing developers at Microsoft. The main method to collect the data was interviewing, which is quite time consuming. Researchers like Rahman and Roy (2014) will be able to collect more detailed data on the actual progress of pull requests, instead of just data on the results, because they can see what developers do while reviewing the pull requests. Jürjens et al. (2005) could have used Octopeer to collect data on the time spent per file in which bugs were found.

Octopeer for Developers

Next to the scientific use that Octopeer has, it also is useful for the developers themselves. Developers are able to see how much time they spend on their review process and how they behave after several other developers have already looked at a pull request. Being able to see their progress over time encourages the developers to better themselves even more. Octopeer can also encourage developers to think about their own pull requests. Some developers will receive a lot of comments on their pull requests before they are merged into the repository. By looking at their previous pull requests, the developers can avoid their previous pitfalls.

Before developers used pull requests to add their own code to a repository, there was another kind of code review as described by Mäntylä and Lassenius (2009). This review required the developer to meet with a number of other developers on the team. During this review the other developers ask questions about the new code and try to find mistakes in it. To make these meetings actually useful, the other developers had to read the code before the meeting, but they would not be able to comment on it immediately. The comments had to wait until the meeting, which means that a lot of time was wasted. The pull request based peer review is much more efficient and less cumbersome, which is why it has become the dominant peer review model (Bacchelli et al., 2014a).

In the remainder of this report we will describe what you can use Octopeer for, in which areas Octopeer extends the existing tools and how long it will take to develop Octopeer.

What Octopeer can do for you

The tool ‘Octopeer for Bitbucket’ aims to provide users with meaningful feedback regarding their review behaviour on pull requests in comparison to other developers. By providing this type of feedback, the tool should enable users to identify their shortcomings when peer reviewing code, and help them to act on these points of attention. Users will then in turn be able to see their progress, when they improve their code reviewing practices.

In order to provide Octopeer users with meaningful feedback the tool measures the browser activity of users when they are reviewing code. This includes, but is not limited to, the tracking of:

- *The time users spend on reviewing a pull request (PR) in total.*
- *The time users spend on reviewing individual files in a PR.*
- *Which elements are in view, and for how long.*
- *Over which elements the user hovers with the mouse*
    - *Buttons*
    - *Code File*
    - *Comment box*

- *The amount of comments a user makes on a PR*
- *PR meta-data such as:*
    - *The amount of commits in a PR*
    - *The number of changed lines of code (LOC) per file in a PR*
    - *The amount of users that already approved a PR*

All of the data that Octopeer collects is stored in a database and can then be analyzed in order to provide users with feedback regarding their review behavior.
Octopeer can provide statistics that vary from straightforward data points, such as the time a user spent on reviewing a PR in comparison to their co-reviewers, to more sophisticated statistics such as the time users spend on reviewing PRs in relation to the amount of times that a PR has already been approved by other reviewers. If, for instance, it turns out that the effort reviewers spend on a code review is greatly reduced after a PR has already been approved for two or more times, it might be wise to drop a third code review as a requirement for a merge. Alternatively new methods can be adopted to ensure that a third review on the same PR is also carefully executed.

Octopeer can also do great work when analyzing additional web pages outside of Bitbucket. Users may leave the PR tab, but are they still considering the PR? Octopeer can partially answer this question by analyzing other tabs in the browser. When a reviewer is executing related search queries in Google, or when he is visiting Stack Overflow, it is likely the reviewer is still working on the PR. When tying these activities to certain files within the PR, we can identify difficult sections in the code, that required the reviewer to conduct additional research.

The focal user

Octopeer targets three types of users: individual developers, software engineering researchers and companies.
As researchers want to store as much data as possible and companies prefer keeping their information hidden, Octopeer allows users to change their privacy settings for themselves. The user can indicate what kind of information should be tracked and whether it should be anonymized or not. These settings can be changed during runtime, which offers optimal flexibility to the user. The focal user of Octopeer will be an individual programmer. This means that a single programmer should be able to use the tool to improve his code review practices. To clarify how a programmer may want to use Octopeer we have created a persona:

“Bob is a 36 year old software engineer who works at a moderately sized software company. He and four of his colleagues work together in a team using an agile development model. Bob is pretty satisfied working in week long sprints as this is a balanced way of developing software. His company recently started using Bitbucket as code repository, and started using pull request instead of the code inspections they used to work with. Bob is glad they did that, because it means he doesn’t have to waste time sitting in an office with five others looking at deprecated code as Bob works faster on his own.

However, Bob is wondering if he is being effective at using the pull requests, and reviewing the code before merging. They used to look at code with everyone in one room before, and now he feels that he has more responsibility. That is why Bob wants to start using Octopeer. He wants to measure what he is doing when reviewing a pull request, and whether he is being effective or not. He hopes the tool can offer insights and tips as to which practices to change. If the tool can help Bob, maybe he can help his team in turn!"

Another potential group of users of the tool is are researchers as mentioned above:

"Janine is a 49 year old data analytics specialist. She is trying to research the behaviour of developers when it comes to git processes. By analyzing these processes, she tries to spot bottlenecks in the process, so that git or related tools may be improved. She hopes that she can use the data Octopeer collects to gain more insight into some of the practices developers have. Especially the data about what developers see on screen, and for how long, that is collected by Octopeer has piqued her curiosity."

Difference with competing tools

Not many tools exist with the singular goal of collecting data about pull requests. Existing tools collect data about an entire code repository or collect very little data about the pull requests at all. What is unique about the tool Octopeer, is that it focusses mainly on pull requests and related data.
Upsource is an example of an existing polyglot code review tool which assists people with performing their code review. Besides assisting with the code review process it also offers statistic analysis.
Upsource is concerned with the following data[1]

- Amount of comments per review
- Amount of reviews per reviewer
- Amount of reviews over time

Octopeer goes further. It collects much more data related to pull requests, which includes:

- Mouse movements, clicks and hovers during the pull request
- Keystrokes during the pull request
- Comments regarding the pull request
- Who reviewed the pull request
- Time taken for reviewing the pull request
- Amount of commits in the pull request
- Additional sources which were used during the pull request

[1] https://www.jetbrains.com/help/upsource/2.0/analytics_main.html

Octopeer does not only collect data, it also offers some simple statistics on the data collected. The statistics can be extended by the use of a special data analytics tool which is being developed for Octopeer. Another competitor is GHTorrent[2] which is a project that collects all available data from GitHub and stores the raw data in a MongoDB database while extracting their structure in a MySQL database. It takes all available data, but it focusses only on GitHub just like Upsource, while Octopeer focusses on BitBucket.

Another difference is that GHTorrent is only collecting the data, for researchers to do with as they please. Octopeer offers simple statistics and options to extend it with a special data analytics tool to make full use of all the collected data.

The data we collect from users is aimed at the improvement of the code review process. For this data from the individual developer or from a group of developers is needed. GHTorrent collects all the data in bulk which makes individual use of it as good as impossible.

Octopeer collects more user specific data than existing competitors, it offers simple statistics on the data with the possibility of flawlessly extending it with a data analytics tool which is under development. While competitors focus only on the GitHub platform, our version of Octopeer is focussed on the BitBucket platform, which gives the possibility of collecting more data than others.

When using Octopeer the user has total control over the types of data that are being collected, by choosing per kind of data whether the user wants to collect it or not, and whether the user wants this type of data to be collected anonymized or not.
In the end, Octopeer will be made available as an open-source project, which gives the user the possibility of extending the tool with features they would like to have.

Besides the version of the tool we create, there is another group of developers in our research team working on another version of Octopeer. The main differences between the two versions are the platforms they can be used on. As the other team is focussing on collecting data from GitHub, our version is collecting data from BitBucket, a platform which provides code reviewers with more options to support their workflow. This means that our tool will be collecting additional data regarding code review process.

Target timeframe and budget

‘Octopeer for Bitbucket’ is planned to be designed and implemented by a team of five software engineers in a timespan of ten weeks. The final product will be delivered on the hard deadline by June 17th. The development is part of the ‘Context project’ where each developer works on

[2] http://ghtorrent.org/

for approximately 28 hours per week. There are also other activities involved in this project so each developer has roughly 20 hours a week that can be invested in the development of the tool. This adds up to a total of 20 * 10 * 5 = 1000 hours that can be invested in the development over a period of ten weeks.

The costs for this tool are nonexistent because all of the work is being done by students.
In a business setting however each developer would receive an hourly rate somewhere in between €30,- to €45,-. This adds up to a total estimated market value from €30.000,- to €45.000,- for the tool.

Reference List

Bacchelli, A., Beller, M., Juergens, E., Zaidman, A. (2014a). *Modern Code Reviews in Open-Source Projects: Which Problems Do They Fix?* CQSE GmbH, Duitsland. Delft University of Technology, Nederland.

Bacchelli, A., Gousios, G., Mukadam, M. & Rigby, P.C. (2014b). *A Mixed Methods Approach to Mining Code Review Data: Examples and a replication study of multi-commit reviews.* Delft University of Technology, The Netherlands & Concordia University, Montreal, Canada.

Van Deursen, A. (2009). *The Software Engineering Group*. Retrieved from http://swerl.tudelft.nl/bin/view/Main/SoftwareEngineeringResearchGroup . Accessed: April 27, 2016.

Jürjens, J., Koller, C., Trischberger, P., Wagner, S. (2005). *Comparing Bug Finding Tools with Reviews and Tests.* Garching, Duitsland. Institut für Informatik, Universität München, Duitsland

Mäntylä, M. V. & Lassenius, C. (2009), What Types of Defects Are Really Discovered in Code Reviews? *IEEE Transactions On Software Engineering, 35 (3), 430-448.*

Rahman, M.M., Roy, K.C. (2014). *An Insight into the Pull Requests of GitHub.* University of Saskatchewan, Canada.