TI2806: Context Project

Tools for Software Engineering

Product planning

Octopeer Analytics

Group: BussInfraManDevOps

Borek Beker (4118650)
Marco Boom (4393031)
Leendert van Doorn (4373286)
Ahmet Gudek (4307445)
Daan van der Valk (4094751)

Supervisors

Context coordinator	Context T.A.	Software Engineering T.A.
Alberto Bachelli	*Aaron Ang*	*Bastiaan Reijm*

May 4, 2016

1 Introduction

The Octopeer project focuses on creating an analysis tool for peer reviews on revision management systems such as Github and Bitbucket. The goal of the project is to track a users actions throughout the peer reviewing process and eventually present this information to the user through visualisations in an easily understandable and highly informative matter. With this data, the users will be able to gain insights on their peer reviewing routines and will have the ability to improve these through time.

The project is a ten week long effort with three teams of five developers working on different parts of the full product, two teams on data gathering and one team on data visualisation. This document contains the description, requirements and other information required for the creation of Octopeer Analytics, a client-side software responsible for the visualisation of the acquired peer review data.

2 Product

The software we are creating will have to adhere to certain requirements and desired features. In this section, a high-level product backlog is given. This backlog lists features the software will or might have at the end of the project. Also a roadmap is created in which our release goals and schedule are depicted.

2.1 High-level product backlog

For the definition of the product backlog, the MoSCoW method is used.This is a prioritisation technique in which product features are divided into four groups. 'Must have' for features the product must have at the end of the project to be considered a product. 'Should have' for features that should be present in the final product to be considered successful. 'Could have' for features that might me implemented if the time allows and if the team wants to implement it. And finally 'Would have' for feature we would like the product to have, but likely won't.

2.1.1 Must haves

The application must have:

- An accessible page.
- A connection to the back-end with the user's data.
- A visualisation of the data that is received.
- A stable implementation which does not crash or come into an erroneous state.

2.1.2 Should haves

The application should have:

- The mining of data that is received from the back-end.
- Different categories for visualisations of the data.
- A tour-like option to learn the know-how of the application.
- A use-able and user-friendly experience throughout using the application.

2.1.3 Could haves

The application could have:

- Only viewing visualisation of the desired data category
- The possibility to change the visualisation type of a certain graph or chart

2.1.4 Would haves

The application would (,but likely wont) have:

- A connection to the Github or Bitbucket API to retreive more personal information.

2.2 Roadmap

Having in mind the items from section 2.1, the development process can be divided in four categories, information gathering, data acquisition, interface design and data visualisation.

In the first two weeks of the project, we will be focusing on information gathering. This means that we will be doing research on subjects such as our target audience, the goal of the project, what data we need and what the user needs of the product. This can be done by brainstorming, reading research papers or approaching potential users in the form of a survey or with interviews.

In the two weeks following that, we will work on completing the UI design and implemented the architecture and data acquisition. The remaining time can than be used to create all kinds of visualisations and coupling the different components of the Octopeer project. The product will be completed at the of the ten weeks.

3 Product backlog

This sections contains a description of the software's (current) features and know-how acquisition in the form of user stories. Additionally, an initial release plan is included.

3.1 User stories of features

As a user I want :

- A portal to see data on peer reviews.
- This portal to acquire and visualise my peer review data.
- Insights on the peer reviews I have completed, such as the:
  - number of peer reviews.
  - length and review times.
- The ability to see my behaviour during peer reviews, such as the:
  - amount of time I spend on a peer review.
  - amount of time I spend on a peer review when code is not pulled to the master branch.
  - amount of time I am concentrated on the peer review.
  - amount of time I am not concentrated on the peer review.
  - speed of and concentration on a peer preview in relation to its length.
  - speed of and concentration on a peer preview in relation to its contributors.
- To know my shortcomings by knowing the types of code that require me to gain extra information through the Internet.
- The ability to asses the quality of my peer reviews by knowing the:
  - amount of communication I require with the owner of the code before approving it.
  - number of times I do not approve changes and whether I let owners of the code know why.

3.2 User stories of know-how acquisition

As a user I want :

- To learn how to use the application by:
  - having the option to take a tutorial or to take a tour throughout the application
  - receiving positive or negative feedback based on my behaviour.
- To be properly informed when the application crashes or comes into an erroneous state by:
  - receiving a clear description of what the error is
  - having the option to send an error report to one of the engineers
  - having my state saved one way or the other within the application (if applicable)

3.3 Initial release plan

The project is developed with SCRUM, an agile software development framework with an incremental and iterative work approach. The development is split in multiple weekly sprints with a deliverable at the end of each sprint. We will call these milestones. We expect the following functionalities to be implemented at the end of these milestones :

3.3.1 Milestone 1 & 2

- A front page is created that can be accessed by the user.
- Demo visuals are created with dummy data which will be replaced with actual user data in the following sprints.
- Interviews are held with software developers to gain information about what they would like to see in a software like Octopeer Analytics.
  - What data would they like to (not) be tracked.
  - What information should be conveyed from the acquired data.
  - What types of visualisation would they (not) like to see.
- The list of information to be visualised is completed.

3.3.2 Milestone 3 & 4

- A backbone architecture is created.
- A connection is established to the database with peer review data.
- Mimicked data is inserted into the database and in turn used for the creation of visualisations.
- GET requests are implemented for data acquisition from the database.
- Simple visualisations completed.

3.3.3 Milestone 5 & 6

- More visualisations are created.
- More complex visualisations are thought of and implemented.

3.3.4 Milestone 7 - 10

- All visualisation aspects are complete.
- Octopeer Analytics is integrated with the two other parts that collect data.

4 Definition of Done

When a certain aspect of the project is completed by the project team, it will be placed under the "Done" category. The following rules apply for the tasks in this category :

- When a software feature is done, the feature has been coded and tested with unit tests and/or UI tests to a degree deemed acceptable by all team members.
- When a software feature is done, the code has been commented and well documented.
- When a software feature is done, it has been reviewed and accepted by at least two other team members and merged with the main code by a third.
- When a software feature is done and thus has been merged with the main code, the new version of the code has passed all new and previous unit tests.
- When a document is done, all required questions have been answered.
- When a document is done, it is checked structurally and grammatically by at least one project member that has not contributed to the document.
- A sprint is considered done when all tasks are done according to the points above, or for all uncompleted tasks a valid technical reason is given for not completing it.
- When a sprint is done, all identified bugs on the products of the previous sprints have been fixed.
- The final product is considered done when the main features are implemented and no technically viable requirement remains uncompleted.

5 Glossary

5.1 Peer review

Peer reviewing is an act of checking the code of a peer before adding this to the main product. Github and Bitbucket are two online revision management systems that allow for peer reviews through the use of pull requests. When someone writes new code or makes any changes to the existing code, they upload it to this system on a different "branch" than where the main code resides. These changes can then be reviewed by others before "merging" the current branch with the main branch if the code is good enough, or commented on if other changes are required.

5.2 Revision management system

A revision management system is a system where different versions and changes between these versions are kept. When used in software development, this gives developers the ability to see the changes to the code through time and even revert changes when needed.

5.3 Product backlog

A product backlog is a list of to-dos to a product. They contain specific tasks for the upcoming days or weeks, while tasks toward the end of the product's development are described more vaguely.

5.4 UI

A UI (User Interface) is the part of a software with which the user interacts as to control the software. It generally consists of sections on which the user can click, hover or slide over to make something happen and sections that show information.

5.5 SCRUM

SCRUM is a software development framework for managing product development. A development team using SCRUM works with fixed-length sprints with deliverables after every sprint. At the start of a sprint, the team decides a work plan with the use of the product backlog for the current sprint. At the end of the sprint, the team needs to have completed the sprint tasks so that the next sprint can build upon the product created at the end of the current. Short sprints such as these give the development team the ability to validate the current works to see if they are still on the right track and if the product is still what the product owner desires.