Product Planning Document

Course: TI2806

ContextProject: Tools for Software Engineering

The Rubber Duck Debuggers (GitHub Division)

4 May 2016

Maarten Sijm
Robin van Heukelum
Mathias Meuleman
Mitchell Dingjan
Youri Arkesteijn

SE TA: Bastiaan Reijm
Context TA: Aaron Ang

Context-Teacher: Alberto Bacchelli

Table of Contents

- Table of Contents
- Abstract
- Introduction
- 1. Product
  - 1.1 High-level product backlog
    - 1.1.1 Must haves
    - 1.1.2. Should haves
    - 1.1.3. Could haves
    - 1.1.4. Won't haves
  - 1.2 Roadmap
- 2. Product Backlog
  - User stories of features
- 3. Definition of Done
  - 1. Bugfixes
  - 2. Document construction
  - 3. Implementing features
  - 4. Research and subsequent decision making
  - 5. Sprints
- References

Abstract

In this document we have described the main overview of the product and the process towards this product. Also the user stories and our definition of done can be found in this document. If we can execute this plan, we think Octopeer will be a success.

Introduc(k)tion

On October 1st, 2013, the US government released the infamous website known as healthcare.gov[1]. Immediately after the release, the website was plagued by security risks, users ran into connectivity problems, the site could not handle the immense pressure and was soon inaccessible. A special team consisting of world's most experienced software engineers had to be gathered in order to get the website fixed, which cost a lot of money, as well as time.

To make sure software engineering projects are done in a better way, scrum methodologies are used to organize these projects. Since it has been proven to work in this team, the scrum methodology will also be used in this project.

In this document the main overview of the product is described. In the first chapter, the product will be described. The main features are highlighted and the Roadmap towards the delivery of these features is given. Chapter 2 contains the user stories that accompany the features. Our definition of done is described in chapter 3.

1. Product

Our product will contain some high-level features which will be named and explained in the first section. In the second section a preliminary overview of the sprints and the features that belong to them. Please keep in mind that this is only an overview of the high-level features. The user stories can be found in the next chapter.

1.1 High-level product backlog

For the Octopeer extension, the requirements regarding functionality and service are grouped under the Functional Requirements. Within these functional requirements, four categories can be identified using the MoSCoW model[1] for prioritizing requirements: Must haves, Should haves, Could haves and Won't haves.

1.1.1 Must haves

- The extension will start logging a user's actions when they are looking at a Pull Request on GitHub.
- A number of user-triggered events will be monitored and saved (in a database)
    - Mouse-clicks (left, right, scroll)
    - Mouse-movement
    - Mouse-hover
    - Keystroke
    - Text Selection
    - Tab change
    - HTML-elements within the view window
- The data will be stored in a database through the use of the provided RESTful API.
- The performance of the extension cannot decrease the performance of the browser by more than 100ms so that it is not noticeable for the average user.
- The extension will have an option menu in which features can toggled.
    - The feature will have an option on anonymizing data (i.e. GitHub name and repository name will be obfuscated by a hashing function).
    - The feature will have options to indicate which types of tracking are allowed.
- The extension will deduce new information from the data that is retrieved.

[1] http://en.wikipedia.org/wiki/MoSCoW_method

    - User specific events that the person is working on during a pull request
    - Data that can later be used in combination with other data
- The icon of the extension should indicate whether the extension is recording

1.1.2. Should haves

- The extension will come up with tips based on the retrieved and deduced data that might be interesting for the users to take them into account.

1.1.3. Could haves

- The extension could have a point-system (with rewards).
- The extension could have a login mechanism.

1.1.4. Won't haves

- The extension won't have an eye-tracking method for deducing information of where the user is looking at.
- The extension won't deduce data from pressure sensors of any kind.

1.2 Roadmap

- Working version (v0.1): This is just a chrome extension at its simplest form. This also includes all research regarding the IDE, Static Analysis Tools etc, including the setting up of these tools by all developers.
- API-bridge: This includes everything that is necessary to connect with the SQL Server that is used to store data.
- Tracking: This will be the collection and storage of the first syntactic[2] data.
- Translation to semantics: This will contain the translation from the gathered syntactic data to semantic[2] data.
- Extended tracking and finalizing product: This includes all additional features regarding tracking. Also the finishing touch for the GitHub part of Octopeer will be done here.
- Integrating with BitBucket Team and Visualization Team: This will require cooperation with the other TSE groups in order to integrate the different parts into one system.

[2] Syntactic data consists of all raw data that can be gathered from GitHub. This data will then be translated into so-called semantic data that is equivalent for both GitHub and BitBucket. This way the Data Processing and Visualization team can work with a single dataset.

These items are all displayed in Table 1 (see below) with their corresponding subtasks from the high-level backlog in section 1.1 and the planned week of completion. For reference, week 1 starts at Monday 18th of April 2016.

High-level feature		**Week**
Research and first working version		2
API-bridge		3
	The RESTful API has to be designed	
	The data will be stored in a database through the use of the provided RESTful API.	
Tracking		4
	The extension will start when a user logs in to GitHub.	
	A number of user-triggered events will be monitored and saved (in a database)	
	The extension will have an option menu in which features can toggled.	
Translation to semantics		5, 6
	The extension will deduce new information from the data that is retrieved.	
Extended tracking and finalizing product		7
	The performance of the extension cannot decrease the performance of the browser	
	The icon of the extension should indicate whether the extension is recording	
Integrating with BitBucket Team and Visualization Team		8-10
	The extension will come up with tips based on the retrieved and deduced data that might be interesting for the users to take them into account.	

Table 1: Roadmap of High-Level Features

2. Product Backlog

User stories of features

We define the user and developer as:
The *user* is the person that actually *uses* the system and of whom the data is collected.
The *developer* is the person that *develops* the system according to the requirements of the user.

As a user
I want the extension to start recording data as soon as I open GitHub

As a user
I want the extension to monitor my behaviour while I am reviewing code
So that I can benefit from it, as the data will be used for possible improvement advice

As a user
I want the extension to provide me with advice on how to improve the way in which I review pull requests on GitHub
So that I can learn from it and become a better reviewer

As a user
I don't want the extension to influence the performance of my browser.

As a user
I want the data that is collected from me to be safe and private
So that no one else is able to use it for any other purpose than obtaining pull request advice

As a user
I want to get advice that is useful from the extension
So that I can actually improve my review behavior

As a user
I would like to have an option menu in which I can put my preferences with respect to privacy and scope of logging of the extension

As a user
I would like to have that the extension shows me when it is active
So that I am aware that my pull request behavior is being recorded

As a user
I would like to know how to use the Octopeer extension

As a developer
I want to collect sufficient and diverse data
So that I can deduce reliable information and useful advice

As a developer
I want the collected data to be stored in a persistent manner.

As a developer
I want to be able to design (a part of) the API that will be used.

As a developer,
I want the extension to deduce information from the gathered data.

As a developer
I want to document certain design choices and the vision of the project
So that the decisions that are made are clear

As a developer
I want to perform researches on possible useful tools that I can use throughout development
So that the I will create certainty that the tools that I'm using make the system more reliable

As a developer
I want to start developing the system in a systematic way
So that I'm assured important aspects are not overlooked.

As a developer
I want the user to be notified when the extension fails
So the user can either reset the extension or contact the developer team

3. Definition of Done

This chapter elaborates on when we, as a team, define a task to be ‘done’. The types of the tasks are listed alphabetically.

1. Bugfixes

An activity that involves bug fixes is considered as ‘done’ if

- the bug has been fixed, according to the person that is responsible for the activity.
- the absence of the bug involved is ensured by additional tests
- the Pull Request of the bug fix has been reviewed by at least two peers before merging
    - any comments on the PR should have been resolved by the person responsible for the bug

2. Document construction

An activity that involves constructing documents is considered as ‘done’ if

- it at least meets the criteria (in terms of an available template or guidelines) that have been provided for it
- the rubrics have been checked if the expectations are met for the final version of the extension (done by the person responsible)
- the document has been spell-checked and grammar-checked

3. Implementing features

An activity that involves implementation of features is considered as ‘done’ if

- the feature has been implemented
- the feature has been sufficiently tested (at least 75% of meaningful[3] line test coverage),
- the feature has been documented
- if the feature included a change in architecture, the architecture document should have been updated
- the code of the feature has been analyzed with static analysis tools
- the Pull Request of the feature has been reviewed by at least two peers before merging
    - any comments on the PR should have been resolved by the person responsible for the feature

[3] Meaningful means that the tests actually test the functionalities of the extension and for example do not just execute the methods involved

4. Research and subsequent decision making

An activity that involves research is considered as ‘done’, if

- the results of the research have been fully documented
    - using arguments, conclusions and provision of relevant information and/or citations
- there is one unambiguous decision made by the group in the conclusion of this research
- the research includes at least 2 other methods to solve the problem
    - In case that there is only one alternative, the research is considered as done if this alternative has been documented, including a comparison of the features between the two solutions.

5. Sprints

Sprints in general are considered as ‘done’ if

- the tickets of all features that have been selected before the feature freeze[4] have been closed
    - Tickets can be dropped before the feature freeze if either an unexpected problem arises or there is not enough time to finish them
    - Features that have not been completed have been documented in the Sprint Retrospective
- the retrospective document of the sprint has been completed

References

[1]: https://en.wikipedia.org/wiki/HealthCare.gov

[4] We put a feature freeze in place on the Thursday afternoon, meaning that nobody can start working on a new ticket. This is to make sure we have a working version by the end of Thursday to show in the meeting on Friday.