Product Vision

by

BlueTurtle

Team Members:

Name:	**StudentID:**	**E-mail:**
Boning Gong	4367308	boninggong@yahoo.com
Clinton Cao	4349024	C.S.Cao@student.tudelft.nl
Michiel Doesburg	4343875	M.S.Doesburg@student.tudelft.nl
Sunwei Wang	4345967	S.Wang-11@student.tudelft.nl
Tim Buckers	4369459	TimBuckers@gmail.com

SE TA:
Bastiaan Reijm

TU Delft

1.1 Who is going to buy the product? Who is the target customer?

Other software developers are going to buy this product. They could use this product to get an overview of the quality of their code. Supervisors of software teams could also use this product to get a quick overview.

Main customers:
Andy Zaidman (Associate professor in the Software Engineering Research Group)
Moritz Beller (PhD on TestRoots[1] in the Software Engineering Research Group)

1.2 Which customer needs will the product address?

The main need we address is improving the software development process. We will do this by analyzing their code to detect locations of weak or bad code. A software developer doesn't want bugs or bad code, but weak/bad code is sometimes inevitable. For large projects it's not feasible that everything is fully perfect and efficient. There is always some kind of technical debt[2]. The consequence is that already existing bug finding tools will produce an incredibly long list of warnings. This makes it hard for big projects to use these tools later on. So there is a need for better bug finding tools, which probably also take the development of code into account.

Questions we could be asking:
Is the code quality improved in a specific period of time?
When there is a cluster of bugs where are they located?
Which new developments and/or changes lead to low quality code?

1.3 Which product attributes are crucial to satisfy the selected needs, and therefore to the success of the product?

The product should give a clear visualization as result. The visualization should not only be clear, but it should also be meaningful. There are certain attributes needed, depending on the different use cases of the users.

For the best usage for the customer the visualization should generate within a reasonable amount of time and should be easy to interpret. Supervisors want to be able to see or measure the quality of the development quickly. Software developers also want constant feedback of the quality and progress they make. So they can not only adjust and improve their code, but also prevent making the same mistakes in the future.

The tool should show the developer correlations between code qualities and bugs. So that developers can gain insight in their own code and can improve the code/their programming skills.
Example questions we want to answer:
Does long methods cause more bugs?
Which couplings and cohesions are there in the code and what effects do they have on bugs/failures?

---

[1] http://www.testroots.org/
[2] https://en.wikipedia.org/wiki/Technical_debt

1.4 How does the product compare against existing products, both from competitors and the same company? What are the product's unique selling points?

One unique selling point is the global overview of bugs.
Another unique selling point is the capability of comparing the quality of your code in different periods of time.

There are already existing products that are similar. But most products address only one problem. Sometimes the product only focus on visualizing the structure of the code. They show cohesion and coupling in the project. On the other side the products show a long list of bugs/weak quality code. We want to combine these aspects into one product.

Existing comparable products on bug finding:

- PMD, Checkstyle and FindBugs: Static analysis tools that we will be using to gather information about bugs/bad code. They have their own mini visualization, but these are constrained to the IDE in which they are used. We want to combine the results of multiple SATs (Static Analysis Tools) and combine this in a good clear visualization.

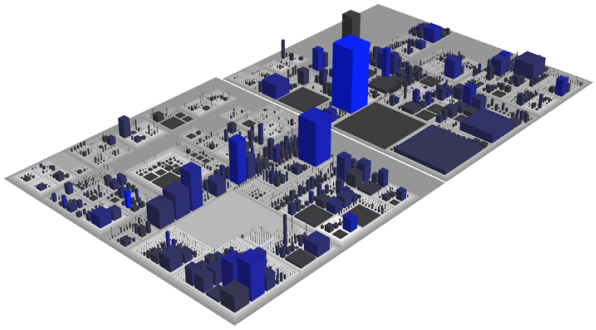
Existing comparable products on visualization of the project:

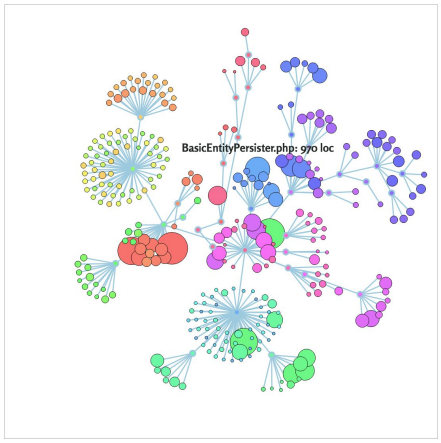
- *CodeCity*: Software that will visualize your java code [1]. It gives an overview of long classes, cohesive classes and more features. Its selling point is their visualization that forms a city where you can 'walk' through. It's not that useful but super trendy.
- *CodeFlower*: Visualizing repository structures and sizes [2]. It makes an interactive tree of your files and of how they are grouped. Files with lots of lines of code are given a bigger circle than smaller files. It works for every Github repository.

Our goal is to combine the features of both categories. To achieve this we are going to use already existing bug finding tools and combine this with visualization frameworks. This should all be integrated into one final product.

1.5 What is the target timeframe and budget to develop and launch the product?

We have ten full weeks after the meeting to develop and present the product. The budget is not discussed, but we don't expect to have any additional expenses.





[1] Example of CodeCity visualization

[2] Example of CodeFlower visualization

http://mozaicworks.com/blog/measuring-code-quality/

http://www.redotheweb.com/CodeFlower/