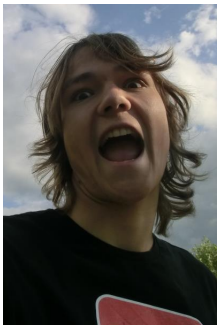
Contextproject Programming Life
Group 2 - GEVATT
Product Planning

TU Delft



Ruben Bes
rbes
4227492

Mathijs Hoogland
mhhoogland
4237676

Jasper Denkers
jdenkers
4212584

Robbert van Staveren
rhvanstaveren
1527118

Willem Jan Glerum
wglerum
4141040

May 22, 2014

Abstract

This product planning gives an extensive overview of the features that the application will have, mainly following the MoSCoW method with Must, Should, Could and Would have elements. These will all be backed by their relevant user stories, explaining them in more detail. We will also give the definition of done, indicating when a feature, report or sprint is truly done.

Contents

1 Introduction

We are group two of the Programming Live Contextproject, a second year course of Computer Science at Delft University of Technology. We will develop a tool for clinical geneticists, which will help to determine the diseases of a person by analyzing their DNA. The name of our tool is GEVATT, which is an abbreviation for GEnetic Variations Analyzer Through Triodata. This tool will visualize the mutations in DNA, triodata and gene interaction. The product will be intuitive to use and will be able to handle existing data to compare the results to. The goal is to develop easier to use software than currently at hand within a fixed time frame.

2 Product

2.1 High-level product backlog

This section describes the desired features of the final application according to the MoSCoW method [1]. These features are categorized into four groups:

- **Must have**

  Requirements that must be satisfied in the final version

- **Should haves**

  Requirements of high priority that should be included if possible

- **Could haves**

  Requirements that are considered desirable but not necessary

- **Would haves**

  Requirements that stakeholders have agreed will not be implemented in the final version, but could be added in the future

2.1.1 Must haves

- Visualize the trio data of the father, mother and child
- Visualize mutations
- Visualize gene interaction
- Looking for known disease mutations
- Retrieving data from existing genetic databases
- Reading VCF-files
- Uploading VCF-files to the server
- Easy to use GUI for doctors, e.g. no redundancy

2.1.2 Should haves

- Save patient data in a database
- Login securely
- Uploading VCF-files to the server in the background

2.1.3 Could haves

- Exporting visualization as raw data which can later be read by the application to reproduce the visualization
- Export visualization as image

2.1.4 Would haves

- Spread computational power over multiple threads, cores, or systems
- Support for mobile web browsers

2.2 Roadmap

This is our major release schema. Because we are using scrum with sprints of one week, the end of every week indicates the end of a sprint. This also means that we will have a working product after each sprint, with new, more or improved features every week.

Sprint 1 10/05-16/05

- Basic setup of framework, read VCF files and connect to database.

At this point, the application will consist of an extremely basic website, without useful code connected to it. In the background we will have developed ways to read VCF-files and establish a connection to the database. This sprint focuses mainly on setting up the project.

Sprint 2 17/05-23/05

- Make useful query's for the database and find all mutations in the VCF file. Also set up the front end, and look into different kinds of visualizations.

At this point, the application will consist of a simple website, with very basic functionality such as logging in and adding patients. In the background we will have developed code to read VCF-files, analyze them and discover and output the mutations. We will also have built basic queries, which will later be combined to receive more useful information. We will also have taken a quick look into what kind of visualizations we want to display.

Sprint 3 24/05-30/05

- Retrieve data from the database associated with the mutations found in the VCF file and make basic visualizations of the mutations. Also link the code and front end.

At this point, the application will consist of a simple website, with basic functionality such as logging in, adding patients and supplying VCF-files. In the background we will have developed code to read VCF-files, analyze them and discover and output the mutations. We will also have built more advanced queries, which combine data from multiple databases. We will also have decided what visualizations we want, and have built basic variants of them into the application. We now have a functional although basic application.

Sprint 4 31/05-06/06

- Visualize the trio data and genes interactions, as well as showing connections between mutations and diseases. The website will now be able to show more information, as well as more visualizations.

At this point, the application will consist of a website, with functionality such as logging in, adding patients, supplying VCF-files and displaying the made visualizations. In the background we will have developed code to read VCF-files, analyze them and discover and output the mutations. We will also have built more advanced queries, which combine data from multiple databases. We will also have improved our visualizations. We now have a functional application.

Sprint 5 07/06-13/06

- Improve visualizations, as well as ironing out issues.

At this point, our application will be the same, except for the visualizations, which will have been further improved.

Sprint 6 14/06-20/06

- Improve usability of the website.

At this point, our application will be the same, except that it will be easier to use, and might me quicker and more responsive.

Sprint 7 21/06-27/06

- Finish report and presentation.

At this point, our application is finished. If, however, small bugs are found, they will be fixed. This sprint focuses mainly on our presentation and reports.

3 Product backlog

All of our features as user stories

3.1 User stories of features

Must have

As a doctor,
When I have opened www.gevatt.nl
I will be on the log-in page

As a doctor,
When I am on the log-in page and not logged in
And I supply correct credentials
I will be logged in

As a doctor,
When I am logged in
And I click 'Log Out'
I will be logged out

Should have

As a doctor
When I am logged in
And on the 'Patients' page
I can see all my patients and none of my other doctor's patients

As a doctor
When I am logged in
And on the 'Patients' page
And I click on a patient
I will see their mutations

As a doctor
When I am logged in
And on the 'Patients' page
And I click 'Add Patient'
I can add a patient

As a doctor
When I am logged in
And on the 'Patients' page
And I have added a patient
It will be added to the database and show up in the 'Patients' page

As a doctor
When I am logged in
And on the 'Patients' page
And I click on delete
The patient will be removed from the database
And it will no longer appear in the 'Patients' page

As a doctor
When I submit a VCF-file
It will be uploaded in a new tab
So I can continue to browse the site

As a doctor
When I am logged in
And on the 'Visualize' page
I can see a visualization of the mutations of a patient

Could have

As a doctor
When I am logged in
And on the 'Visualize' page
I can store a visualization

Won't have

As a patient
When I have opened www.gevatt.nl
I can log in

4 Definition of Done

This section describes when a feature is really done and ready to be integrated in the system. We will define this for a feature, sprint and the end product.

A feature is finished when it is fully tested and the code is accepted by other developers. The tests should be implemented with JUnit and the code coverage should be high enough. Furthermore, the code must be fully documented with JavaDoc and should match the rules of CheckStyle.

A sprint is finished when the whole application is tested and approved, just like a feature. However the continuous integration system should also accept the build. Next, the developers and users will test the system by hand to check for bugs.

The end product is finished if all the Must- and Should haves of the MoSCoW model are implemented and tested as described above. The reason we are not content with just the Must haves is because only the Must haves result in a very basic application. The product should be approved by the stakeholders based on the looks and feels, and they should be happy with the product.

Furthermore in addition to all this, the code should be well documented, tested, style checked and integrated. This will all be evaluated by the SIG (Software Improvement Group) and should be improved after the first check.

5 Glossary

- VCF

  Variant Call Format, the format of the files containing the mutations.

- JUnit

  Testing framework for JAVA

- JavaDoc

  Documentation of JAVA code

- CheckStyle

  A plugin in JAVA for checking the style of your code

References

[1] Kevin Brennan et al. *A Guide to the Business Analysis Body of Knowledger.* Iiba, 2009.