Product Planning Context Project
Desoxyribonucleïnezuur

Toine Hartman - tjbhartman - 4305655
Martijn van Meerten - mvanmeerten - 4387902
Iwan Hoogenboom - ihoogenboom - 4396634
Yannick Haveman - yhaveman - 4299078
Ivo Wilms - iwilms - 4488466

11 May 2017

Abstract

This document contains the planning for the product. A high-level product backlog is given, which contains the prioritization of features in the form of user stories. Next, a roadmap of milestones is given, to guide us in our development. Furthermore, a detailed product backlog is defined, and finally, the definition of done for certain aspects of the project is given.

Table of contents:

1. Introduction

This document is the Product Planning of the Programming Life context. We will, where possible, follow the planning and guidelines as described.

2. Product

2.1 High level product backlog

Here we will discuss features we want to have in our program using the MoSCoW model. The MoSCoW model makes use of 4 groups

Must haves: These are the features that the program can not go without.

Should haves: These are essential features, but the program will not fail if these features are not part of the application.

Could haves: These are features that are very nice to have but are more quality of life features. They will only be implemented if we have time after implementing all should haves and must haves.

Won't haves: These are features we will not implement because they will take too much time, or add too little.

- Must haves:

The application must be able to load in and parse files that are in a GFA format.
The application must display the different segments in the form of a graph.
The application must be able to visually display separate segments of the genome.
The application must be independent from the platform using it.
The application must be able to zoom in and out on different parts of graph to give a better overview of the data.
The application must be able to load files containing 341 chromosomes and 250 million base pairs.
The application must allow the user to select a node and view up to 10000 nodes around the selected node
The application must allow the user to navigate over the graph by panning.

- Should haves:

The application should allow the user to quickly execute the 10 latest searches.
The application should have a search function for the base pairs in a segment.
The application should allow the user to copy the base pairs in a segment to the clipboard.
The application should have a search function on the text in the annotations.
The application should visually encode segments to display more detail about segments.
The application should allow the semantic zooming.
The graph should use glyphs to simplify bubbles depending on the height of the view.
The application should allow the user to annotate segments of the graph.

- Could haves:

The edges could change of thickness to visualize the amount of genomes that use that edge.
The application could include different types of GUI layouts.
The application could have different color schemes for nodes (change on what criteria the color of the nodes is determined).
The application could have different different color scheme for the visually impaired (colorblindness).
The application could have a way of exporting GFA files.
The application could have a way of editing GFA files.

- Won't haves:

The application won't be portable to mobile devices.
The application won't support plugins.
The application won't have cloud support.

2.2 Roadmap

Milestones are to be completed each week starting from the second week. They will be accompanied with a release at the end of each week. The milestones set to be completed are described below.

Milestone 1: Basics
Basic UI.
Basic graph visualization.
Able to load the 10TB genomes.

Milestone 2: Data scaling
Upgrading the data structure to load 341 TB, tomato and human chr19.
Visualising by center node and radius. With roughly 500 nodes left/right
Being able to move the centerpoint.
Center Point selection layout.

Milestone 3: Layout and Navigation
Layout graphs efficiently without overlapping.
Being able to pan and zoom.

Milestone 4: Visual encodings
Search/filter, conditional coloring of nodes and edges.
Color paths based on genomes using the edge.
Edge thickness based on amount of genomes use this edge.
Edge highlights

Milestone 5: Interval annotation
Bookmarks and gene annotations.
Highlighting paths.

Milestone 6: Semantic zoom
Semantic zooming.
Stacking of multiple overlapping annotation paths

Milestone 7: Eye-candy
Hover over nodes display pop-up with node information.
More to be determined during the later weeks in the project.

3. Product Backlog

3.1 User stories of features

As a user
When I launch the application
I want to be able to load a Graphical Fragment Assembly (GFA) file.

As a user
When I have loaded a gfa file,
And have not yet done anything
A random subgraph must be displayed.

As a user
When I have displayed the graph
I want to be able to zoom in and pan around

As a user
When I zoom in I want the graph to semantically load the parts on which I zoom in.
When I zoom out I want small details to be merged together.

As a user
When I am looking at a subgraph
And it contains a known pattern
I want them to be displayed.

As a user
I want to be able to put annotations on nodes and groups of nodes.
Before I close the program I want to be able to save these annotations.

As a user
I want to be able to quickly execute my 10 most recent search.
So when I want to go back I can do that easily.

3.2 User stories of defects.
As a developer
When a defect is found, I will make an issue and add it to the backlog.
Then I will determine the importance of the bug.
To decide whether I want to fix the bug or develop new features.

3.3 User stories of technical improvements.
As a developer,
When reading code,
Through proper documentation,
I want to be able to easily understand the functionality of methods and fields.

As a developer,
I want the code to be stored on github.
So my whole team can access all the code in parallel.

As a developer,
When I push a commit to git,
I want Travis to run a maven clean test build
So that I know the project runs on a clean machine

3.4 User story of know-how acquisition
As a development team,
When we decide on features in a field we are inexperienced in,
We will read appropriate literature to gain a better understanding of that field

3.5 Initial Release Plan
In our initial release all the functionality of the must haves will be included. Also all the should haves will be included unless they are deemed not worth the effort for the result. The could haves will be implemented if all must haves and should haves are implemented far before the release date.

4. Definition of Done

Here we will give our definition of when a backlog item, sprint and a release is done.

A backlog item is considered done when:
The functionality it describes is implemented.
The code is (where possible) tested to a branch coverage of at least 75%.
The code is peer reviewed and travis CI does not fail.
The feature is merged with the develop branch without any problems.

Sprints are considered done when:
All the backlog items for that sprint are done or not needed to be implemented anymore.
A sprint reflection has been made.

A release is considered done when a working version of the software is released containing all must-have features described in section 2.1.

5. Glossary TODO

Github: A web-based Git or version control repository.
GFA: A file extension for graphical fragment assembly files.
MoSCoW: A model for designing requirements documents. Based on prioritizing features in 4 different categories.
TB: Acronym for tuberculosis bacteria.
UI: Acronym for User Interface.
MRF: Minimum required features.