

# Product Planning

Toine Hartman - 4305655  
Martijn van Meerten - 4387902  
Iwan Hoogenboom - 4396634  
Yannick Haveman - 4299078  
Ivo Wilms - 4488466

4 May 2017

## Table of contents:

1. Introduction	2
2. Product	2
- 2.1. High level product backlog	2
- 2.2. Roadmap	3
3. Product backlog	4
- 3.1. User stories of features	4
- 3.2. User stories of defects	4
- 3.3. User stories of technical improvements	5
- 3.4. User stories of know-how acquisition	5
- 3.5. Initial release plan (milestone, MRFs per release)	5
4. Definition of done (backlog items, sprints, releases)	5
5. Glossary	5

# 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

*needs to be synced with product vision.*

Here we will discuss featured we want to have in our program using the MoSCoW model.

The MoSCoW model makes use of 4 groups

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

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

**Could have:** 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.

**Won't have:** These are features we will not implement because they will take too much time.

- Must have:

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 visually encode segments to display more detail about segments.

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 graph must use glyphs to simplify bubbles depending on the height of the view.

The application must be able to load files containing 341 chromosomes and 250 million base pairs.

The application must allow the user to annotate segments of the graph

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.

The application must allow the semantic zooming.

- Should have:

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 have a search function on the text in the annotations.

- Could have:

The application could include different types of GUI layouts

- Won't have:

The application won't be portable to mobile devices

## 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 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

Edge thickness on weight

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 TBD

## 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

As a developer,  
When I push a commit to git,  
I want Travis to run a maven build.

### **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 in the release.  
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**

A backlog item is considered done when the functionality it describes is implemented and tested (where possible) to a branch coverage of at least 75%. Sprints are considered done when the deadline is reached. 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:**

**GFA:**

**MoSCoW:**