# Product Planning

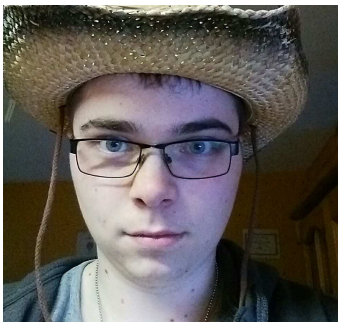
*Context project - Programming life*

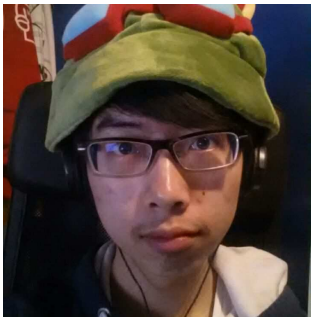
15 May 2015

Justin

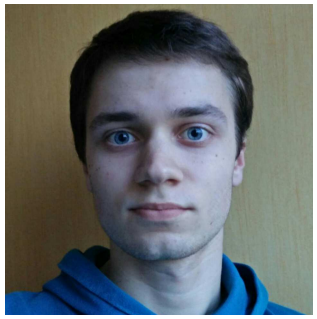
Marissa

Mark





Chak Shun
*Scrum Master*



Maarten
*Product Owner*

***Team PL1***

| | | |
|---|---|---|
| Chak Shun Yu | csyu | 4302567 |
| Justin van der Hout | jrtevanderhout | 4319982 |
| Mark Pasterkamp | mpasterkamp | 4281500 |
| Marissa van der Wel | mmvanderwel | 4323602 |
| Maarten Flikkema | mhflikkema | 4306538 |

# Abstract

This document describes the planning of the project on multiple levels. First, we will decide on which requirements belong to which category following the MoSCoW[1] principle. After that, in the Roadmap, the main tasks for each sprint are described. In the third chapter, the requirements are turned into user stories and the initial release plan will show how the milestones are divided over the weeks. Finally, the Definition of Done describes what we define as done for our features, sprints and releases.

# Table of contents



[1] MoSCoW = Must haves, Should haves, Could haves, Would haves

# 1. Introduction

The Programming Life Context Project 2015 has the goal to develop a new and better genome browser.

# 2. Product

This chapter will evaluate all the customers' requirements for the product. This will be done by dividing them into the four categories of the MoSCoW method: Must haves, Should haves, Could haves and Won't haves.

## 2.1. High-level product backlog

The aim for this project is to create a fully functional genome browser which will visualise the differences between different genome strings. In order to achieve this goal, the product has to provide basic functionalities such as visualising the differences between genomes and semantic zooming, as well as provide more advanced functionalities like analysing the differences between the genome strings in the context of the evolutionary relationship between bacteria.

### Must have

The "Must have" requirements are requirements that must be satisfied in the final product for the solution to be considered a success.

- Ability to interactively explore a sequence graph representing the genome architecture of multiple strains.
- Semantic zoom levels.
- Identification of mutations and determination of the type of mutation.
- Gene annotation.
- OS independency.

### Should have

The "Should have" requirements are requirements that that we plan to implement, but are at risk of being dropped if they turn out to not be feasible.

- A way to load input faster after the first time it is loaded.
- Visual encodings for different classes of mutations.
- The ability to filter on genomes and on mutation classes.
- Put this graph in the context of the evolutionary relationship between bacteria
    - Provide indications for convergent evolution of variants.

### Could have

The "Could have" requirements are requirements that are considered desirable, but not necessarily needed. These will be included if time and resources permit it.
The functionalities that we will implement, should time allow it.

- The ability to manually add labels to nodes of the graph.
- Integrate our program with other resources, such as literature databases, mutation databases, to identify graph features that are interesting for further investigation.

### Won't have

The "Won't have" requirements are features the client would like to have, but has agreed with us that these features will not be implemented.

- This project does not have "won't haves" because the client has already defined very specific requirements for the project.

## 2.2. Roadmap

This roadmap shows which high level tasks we plan to do in every sprint.

| **Sprint** | **Task** |
|---|---|
| 1 | Implement a crude graph visualization<br>Orientate on integrating phylogenetic trees and semantic zoom levels. |
| 2 | Refine graph visualization.<br>Setup simple graphical user interface. |
| 3 | Visualize the nodes using different properties.<br>Create a graphical user interface.<br>Create a semantic zoom level which merges small point mutations. |
| 4 | Implement a graph reader and writer to load imported graphs quicker.<br>Implement the ability to filter the graph on certain genomes.<br>Identify, determine and visualize small mutations. |
| 5 | Create a semantic zoom level that features amino acids.<br>Implement the ability to filter the graph on mutation classes.<br>**Prototype version of the product (SIG).** |
| 6 | Create a semantic zoom level that features genes.<br>Implement gene annotation. |
| 7 | Integrate more information into our program: evolutionary relationship between bacteria (and information from other resources). |
| 8 | **Final release.** |

# 3. Product backlog

In this chapter we will describe the requirements in terms of user stories. This makes more concrete task management and feedback possible. The User stories are classified in three categories: features, technical improvements, and know-how acquisition.

## 3.1. User stories of features

As a user,
I want the program to be usable independent of my operating system.

As a user,
When I import genome data into the program,
I want to be presented a sequence graph representing the genome architecture of my imported data.

As a user,
Given I was presented a sequence graph representing the genome architecture of multiple strains,
I want to be provided gene annotation on the graph.

As a user,
Given I was presented a sequence graph representing the genome architecture of multiple strains,
I want the program to put the graph in the context of the evolutionary relationship between bacteria,
And provide indications for convergent evolution of variants using phylogeny.

As a user,
Given I was presented a sequence graph representing the genome architecture of multiple strains,
I want to be able to filter the graph on certain genomes.

As a user,
Given I was presented a sequence graph representing the genome architecture of multiple strains,
And to be able to filter the graph on mutation classes.

As a user,
Given I was presented a sequence graph representing the genome architecture of multiple strains,
I want to be able to manually add labels to nodes of the graph.

As a user,
Given I was presented a sequence graph representing the genome architecture of multiple strains,
I want to be able to interactively explore the sequence graph.

As a user,
Given I am exploring the sequence graph,
I want to be able to explore it more thoroughly using different semantic zoom levels which provide me different (visual) information.

As a user,
Given I am exploring the sequence graph,
I want the program to provide me with information about where and what type (INDEL, SNP) of mutations occurred.

As a user,
Given I am exploring the sequence graph,
I want to be provided with visual encodings for different classes of mutations (Insertion, Deletion, SNP, Duplication, Inversion, Translocation).

As a user,
Given I am exploring the sequence graph,
I want the program to provide visual representations and encoding of metadata associated with samples, such as drug resistance, location of isolation, isolation date, etc.

As a user,
I want the program to integrate with other resources, such as literature and mutation databases,
So that the program can identify and provide me with graph features that are interesting for further investigation.

## 3.2. User stories of technical improvements (if applicable)

As a user
Given I have started the application and I can see the GUI
When I press the “load graph” button
I want to still have a responsive GUI while the graph is being loaded.

As a user,
Given I already have imported my genome data into the program before,
When I load my data into the program once more,
I want it to be done faster than the first time.

## 3.3. User stories of know-how acquisition

As a user
When I launch the application
It will be easy to understand the data being showed.

As a user,
Given I was presented a sequence graph representing the genome architecture of multiple strains,
I want to be able to see if a mutation is shared by genomes,
And have this put into the context of the evolutionary relationship between the genomes.

### 3.4. Initial release plan

| Week | Milestones |
|---|---|
| 4.2 | Crude graph visualization. |
| 4.3 | Refined graph visualization.<br>The user now has access to a simple graphical user interface. |
| 4.4 | Nodes are visualized.<br>Users now have access to a refined graphical user interface.<br>Point mutations can be merged on a semantic zoom level. |
| 4.5 | Loading previously imported graphs is done quicker.<br>Users can now filter the graph on specific genomes.<br>Small mutations are now identified.<br>The type of small mutations are now determined. |
| 4.6 | Users can now view the graph in a semantic zoom level that features amino acids.<br>Users can now filter the graph on different kinds of mutations.<br>**Prototype version of the product (SIG).** |
| 4.7 | Users can now view the graph in a semantic zoom level that features genes.<br>Users can now view gene annotation. |
| 4.8 | Users are now provided with data about the evolutionary relationship between bacteria (and information from other resources) in our program. |
| 4.9 | **Final release.** |

## 4. Definition of Done

The purpose of the Definition of Done (DoD) is to inform everyone involved in the project about what it means when a certain aspect is done. These aspects can be just features, but also sprints and entire releases. The DoD is defined separately for each of these types of aspects.

### Feature DoD

A feature has a set of user stories. A feature is considered as done when all the functionalities described in those user stories are implemented and tested. Tested means manual/functional tested, unit tested, and acceptance tested. Furthermore the code has to be well commented and documented.

### Sprint DoD

A sprint is considered as done when all the features defined in the sprint plan are considered as done, as defined in the previous paragraph, at the end of the sprint period.

### Release DoD

A release will considered done when the corresponding sprint are considered as done, as defined in the previous paragraph, and the customers are satisfied with the program as a result of the sprint. This means the customers have agreed that the implemented features, as they are implemented by us, are what they wanted it to be and are satisfied with how they are.

# 5. Glossary

| | |
|---|---|
| **GUI** | Graphical User Interface, meaning the actual window and its content the user will see on the screen. |
| **User stories** | High level scenarios describing the user's requirements from the user perspective. |
| **Product backlog** | A list of features containing all functionalities desired in the product. |