Product Planning

Tycho van Heems - tvheems - 4464567 Dex van Leeuwen - dqvanleeuwen - 4475461
Martijn Straatman - martijnstraatm - 4442504
Jochem Tiessen - jtiessen - 4478339
Michael Tran - michaeltran - 4499638

May 11, 2017

Contents

1 Introduction

To help with research on DNA, there are certain tools which can display graphs represent DNA. Currently, there are not enough tools yet to efficiently work with genomes. Genomes contain a lot of information, which makes it easy to miss important details when studying a genome. Especially when comparing multiple different genomes, it may be difficult to keep track of those, let alone retrieve useful information from these genomes.
Our goal is to provide a new tool for studying DNA. We will make a multi-genome browser, which can read in files containing the information of the genomes. This information is contained within a graph. This graph or parts of it should be visualized on screen. The user of the multi-genome browser will be able to retrieve subgraphs which can be zoomed in on and zoomed out of. It will also be possible to pan the screen to see neighboring parts of the subgraph. Another important aspect of the browser will be the ability to lookup genomes and annotate them.
The rest of this document will explain how we (the development team) envisions the product by talking about our customer and what the customer wants. We will also talk about how our product differs from already existing tools. Lastly, the timeframe and budget in which the product must be made will be discussed.

2 Product

In this section, a product backlog and a roadmap will be created to show the needed functionalities and the expectations of the process during the development. The product backlog focuses more around the features that need to be made and the prioritization of those, while the roadmap focuses more around what is expected to be done at what moment. We will also discuss the initial release plan.

2.1 Product Backlog

In the product backlog, we use the MoSCoW method to describe the features we envision to have implemented in the final product.

Must haves:

- The user can select GFA files to be used in the application.
- GFA file parsing. When the user selects a file, that file must be parsed to a graph structure.
- When parsing, the GFA file should be parsed to a data structure that supports efficient retrieval of sequences and center node queries. The sequences must be stored in storage and the graph structure must be stored in memory.
- Stored graphs must be visualized on the screen. Not the whole graph should be on screen at once so a subgraph based on a center node and radius must be shown.
- There must be an efficient layouting algorithm that makes sure the graph layout is not cluttered.
- A GUI must be provided in which the graph is visible, the user can change the current center node and radius and the user can navigate through the graph.
- The application must be platform independent.
- The application must be bug free upon final release.

Should haves:

- Data must be visually encoded within the graph such that the user can use that information.
- Bookmarks, used to return to previously visited nodes.
- Path selection and highlighting.
- Semantic zooming. Used to provide clear visuals when graph structure becomes too complicated.
- The application is easy to use. Nothing should confuse the user in any way.
- The user does not want to wait long when parsing a GFA file. Especially, when the file has already been loaded before.

Could haves:

- A manual on how to use the application.

Won't haves:

- De novo assembly will be fully ignored. However, this might be useful to add in further development after this project.

2.2 Roadmap

The roadmap shows which features are planned to be finished at what moment. For this, milestones are created which specify the minimum functionalities it should have at that point. There could be more features, but there should not be less than specified in this plan. There should be a runnable version of the product at each of the dates.

Milestones:

1. GFA files must be parsed to some data structure. A basic visualization must be visible within a simple GUI.
2. Most important is that all provided GFA files are being parsed within reasonable time. Besides that, instead of the whole graph at once, a subgraph based on a center node and radius should be visible.
3. An efficient layouting algorithm must be present to create a good layout for every graph. Also, the user must be able to zoom and pan within the application.
4. Visual data encoding is top priority this milestone. The user must be able to see consistency of the nodes.
5. Path must be highlighted when found. Bookmarks must be present to return to previously visited nodes.
6. Semantic zooming. This milestone it is important that nested bubbles are visualized in a clearer way when zooming out. Visually complicated structures within the graph should be merged into one simpler node and information about the underlying structure must be provided.
7. The last milestone is focused on providing extra features which make using the multi-genome browser easier and/or smoother for the user.

2.3 Initial Release Plan

The initial release plan describes how we imagine the product to be released. It will cover incremental releases and the final release.
In the roadmap, different milestones are specified which cover what features should be done at which moment in time. At each of those moments, we are supposed to bring out an executable product and demonstrate it during a meeting in front of the customer. The customer can give his feedbak on the current state of the project and he can ensure that the product is moving in the right direction. This means that at every milestone, a incremental version of the product will be released.
The last milestone will cover the final release of the product, which should contain all features in the product backlog including some other bonus features if possible. It should also make sure the product is of a certain quality. During the final meeting, the customer will tell the development team whether he is content with the final product.

3 Definition of Done

In the definition of done, it is defined when a part of the project is done according our standards. A definition is created for features, sprints, milestones and the final product. All these definitions of done specified here, are made according to the definition of done of the development team for this product.

3.1 Features

A feature is considered done when its functionalities are completely working and if the features are tested extensively. For testing, automated tests are written to guarantee the quality of the final product. For features, we will mostly be unit testing, where we consider a feature tested properly when at least 70% of the branches in its code are tested and when every test succeeds.

3.2 Sprints

A sprint will be considered done when all the user stories specified in the corresponding are either finished or when there is a legit reason for it not to be finished, e.g. the specified user story is not applicable to the product anymore. Also, there needs to be an executable version of the product at the end of each sprint, which should be tested extensively. A boundary of 80% branch coverage over all the code was chosen for the code to be considered tested well enough.

3.3 Milestones

Each of the earlier specified milestones will be considered done when all the defined functionalities for the milestone are included in the product. Since these functionalities are features, a testing boundary of 70% branch coverage was chosen as minimum coverage.

3.4 Final Product

The final product will be considered done when the following four requirements are achieved. First of all, the final product should be able to compile and run on the specified operating systems. Second, all features in the must-haves must be in the final product, because when these are not present, the product will not have one of its core functionalities. The specified should-haves should be included as much as possible. However, in contrast to the must-haves, it is not as strict, which means that not every should-have must be included, although it is the preferred result to include all of them. Third, the customer should be content with the final product. If the customer is not content, the production has been for nothing. Last, the whole product must be tested extensively to guarantee the quality of the product and to reduce the number of bugs. As in the sprint, the product is well tested if 80% of its branches are covered with automated tests.

References

"The Scrum Product Backlog - International Scrum Institute". Scrum-institute.org. N.p., 2017. Web. 4 May 2017.

"Agile Development Release Planning". VersionOne. N.p., 2017. Web. 4 May 2017.

"Agile Roadmaps: Build, Share, Use, Evolve The Agile Coach". Atlassian. N.p., 2017. Web. 4 May 2017.