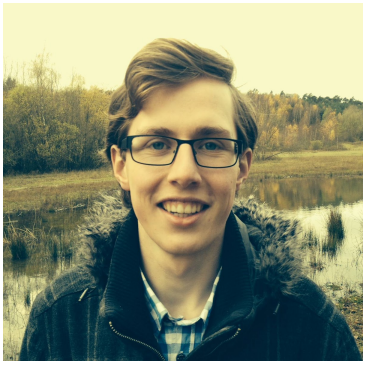
Product planning Genome Explorer

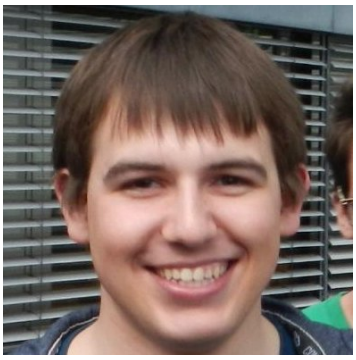
Programming Life: Context Project TI2806

15-5-2015
Group 2
Supervisors: Dr. A. Bacchelli & Dr. T. Abeel

Gerben Oolbekkink (gjwoolbekkink, 4223896)



Jasper Boot (jboot, 1516272)



Jasper Nieuwdorp (jnieuwdorp, 4215796)

Jim Hommes (jimhommes, 4306090)

René Vennik (rvennik, 4102959)

Table of contents

1. Introduction……..[p. 3]

2. Product……..[p. 3]

2.1. High-level product backlog……[p. 3]

2.2. Roadmap……[p. 4]

3. Product backlog ……[p. 5]

3.1. User stories of features……...[p. 6]

3.2. User stories of technical improvements……..[p. 7]

3.3. User stories of know-how acquisition……[p. 7]

3.4. Initial release plan……[p. 7]

4. Definition of done……...[p. 8]

5. Glossary ……...[p. 8]

1. Introduction

This document reflects the goal of our product and the possible shape it will take. It is a guideline, and the user stories that we have defined should be in agreement with the user. The user stories will give an accurate view of the product that we are trying to create.

This document should be read with the product vision in mind. The contrast with the product vision is that this document is more based on the current process of our project team, and our definition of the product.

2. Product

The product should give an interactive visualisation of multiple genome strands. It is essential that the product can visualise large numbers (less than a thousand) strands of DNA with relevant information. This information can be divided into two main groups, namely present information (information that can be deducted directly from the data) and augmented information (information that is available in different sources and can be linked to the data). The focus lays on the first type of information since the second type can be added later.

The product will be used in a diverse but clinical environment and should provide valuable information on every level of detail. The product is made for biologists from with post-graduate master degree and higher: they should be able to understand and use the product.

2.1. High-level product backlog

These are the product items (highest priority first):

- The user should be able to read the graph files
- The user should be able to see the genome graph with edges between the nodes
- The user should be able to see a phylogenetic tree of the genomes
- The user should see a detailed view of the phylogenetic tree
- It should be a program that is easy to learn and efficient in my work.
- The user should have a clear view of relevant information on all levels (semantic zooming)
- The user should have a clear & efficient view of the genome graph
- The user should simply load all relevant files in a certain directory
- The user should see the DNA-sequence that belongs to each node
- The user should be able to navigate between the phylogenetic tree and the genome graph
- The graph should be annotated with information that can be found in the data itself
- The graph should be annotated with information available from external sources

2.2. Roadmap

We have 10 weeks. Every week there is an iteration. At the end of every iteration we will deliver: The sprint plan for the coming iteration, the sprint reflection of the previous iteration, code for a working product and if applicable other documentation.

Week 1: Getting familiar with the project
This week includes setting up our programming environment, studying the background information of this context and getting familiar with the provided info.

Week 2: Setting up a basic implementation and basis
This week includes setting up a basic application, a basic GUI and reading the graph info.

Week 3: Extending the basic implementation and inquiring about needs
Here we received the feedback of our first iterations, and we should implement that feedback.

Week 4: Extend implementation
Extending the implementation with more specific features.

Week 5: Adapt the implementation to the target audience
After gathering info from the target audience and accessing the information we should implement the current implementation to their needs.

Week 6: Extend implementation and add features
Further extension of the implementation.

Week 7: Add features and analyze release environment.
We add features and research how our users will use our program. Here we might have to tweak the user interface to make it more clear for example.

Week 8: Adding features, improving previous work
Nearing the end of the final product, we should improve work where needed and adding the final features.

Week 9: Finalize product
Introducing the last changes, small features and small improvements.

Week 10: Make last changes
In this last week the product should be finished and ready to be shipped. Changes should be final and minimal here.

3. Product backlog

The items in this part are prioritized following the MoSCoW method.

Must have

- Reading graph files and visualizing the genome graphs.
- Reading newick files and visualizing the phylogenetic tree.
- A link between the genome graph and the phylogenetic tree.
- The ability to handle large numbers of genomes.
- A simple way to select all files that are relevant.
- The product should be platform independent.

Should have

- Annotations on the genome from external sources.
- Additional information that can be derived from the information.
- Working on simple consumer hardware (possibly 32-bits)
- On different levels of detail only show relevant information to keep the view clear. (Semantic zooming.)
- Identify and mark different types of mutations in the genome.

Could have

- An smart way to teach users to use the program more efficiently.
- Allowing the user to add it's own annotations to the data.
- A breadcrumb trail used for navigating between the different zoom levels
- Automatically identify convergent evolution.

Won't have

- Export data to common used format to improve usability in research.
- Use of off-site resources(cloud computing for example).

3.1. User stories of features

As a user I want to simply load all relevant files in a certain directory
This indicates the feature of a workspace for example. The user does not want to be bothered with a loading scheme for all of the different files every time. Instead the directory with all the different files should be loaded.

As a user I want to be able to read the graph files
A trivial feature is the reading of graphs. Our product should interpret the graph files and load this in the screen.

As a user I want to be able to see a phylogenetic tree of the genomes
We should display the information of the phylogenetic tree.

As a user I want to be able to see the genome graph with edges between the nodes
We should display the information of the genome graph.

As a user I want to have a clear & efficient view of the genome graph
Displaying the graph comes with problems: because there are so many nodes and edges it may get really messy and unclear. Our task is to show the graph efficiently and to make sure it is clear.

As a user I want to see the DNA-sequence that belongs to each node
The graph should be able to display information of all levels. The nodes are not the only thing, but the exact data should be made available in the product.

As a user I want to see a detailed view of the phylogenetic tree
Just as with the genome graph, the phylogenetic tree should be able to display the exact data.

As a user I want to be able to navigate between the phylogenetic tree and the genome graph
The user should be able to do his research and to make the referencing easier, he should know where in the genome graph he is located in the phylogenetic tree.

As a user I want the graph to be annotated with information that can be found in the data itself
As a lot of conclusions can already be deducted from the information, the product should do this for the user.

As a user I want the graph to be annotated with information available from external sources
The developers should include features from information found in e.g. papers and trusted sources. This information is already made available and the product should implement this information.

As a user I want to have a clear view of relevant information on all levels (semantic zooming)
A lot of information will be displayed in the product, and not everything is useful. So, the information that is shown should be displayed in levels. When the user wants to see the

high-level information, he can zoom in. If he zooms out he will only see the low-level information.

As a user I want a program that is easy to learn and efficient in my work
The users are able to adjust to software but are not software engineers. We have to make sure that the user is able to use the product to its full extent, and should not be confused.

As a user I want to see mutations in the genome marked
Marking mutations, and possible identifying them, takes a lot of work from the user. This is a nifty feature for the product.

As a user I want to see percentages of each base in each node
Seeing the percentages of the bases on a high level is useful for the user. Important is that this is not a primary marking factor, but only available when looking at all the nodes.

3.2. User stories of technical improvements

As a programmer I want to make the product as efficient as reasonably possible to ensure usability on a large scale
Every iteration from iteration 3 will have refactoring in the *sprint plan* to maintain good code quality and thereby scalability. The data structures will be chosen with care and if needed replaced to maintain a low time complexity.

As a programmer I want to make the program run on a lot of different platforms
We program in Java, which has it's own environment and is platform independent. If we stay within this environment the end product will also be platform independent.

3.3. User stories of know-how acquisition

- As a team we should follow all lectures and read all given material
- As a developer I want to understand the area that I work in and read information about this.
- As a programmer I have the duty to read papers about what I'm working in.
- As a designer I have to know who my target audience is and what it´s needs are.

3.4. Initial release plan

We use milestone oriented planning. At the end of every iteration is a milestone and we want to be done with all tasks that were planned for this milestone. If something changes we will evaluate why this changed (e.g. more complex than expected, not as valuable as expected, not reachable) and update the planning accordingly. It is important to have a working version at all times and to constantly prioritize the backlog to ensure an efficient continuation.

4. Definition of done

Done implementing means:

1. The code meets our quality demands:
    - Test coverage > 75%
    - Implements desired feature
    - Maven builds without errors
    - No severe issues reported by: FindBugs, CheckStyle, PMD, CPD and Travis
    - Code is commented
2. When you believe to be done with a task, commit and create a pull request.
3. After 2 other persons reviewed your code and agreed to it, the last person of the 2 merges the branch.
4. The owner of the tasks is responsible for deleting the branch from Github & Closing the issue.
5. Before the end of each iteration there should be UML supporting the code

A sprint reflection is done when all points from the previous sprint plane are reviewed, assigned an actual effort and notes of applicable. Every team member will get a chance to add problems that he encountered during the iteration and there will be a story looking back on the sprint and providing context that all team members will agree on.

A sprint plan is done when all team members feel they have a fair amount of work that they feel adequate to do and when there is no more reasonable room for more tasks.

5. Glossary

1. Genome: the haploid set of chromosomes in an organism.
2. Strands: a reading of the genome expressed as a list of bases (A, C, T and G's).
3. Mutation: an alteration of the DNA, this can be deletion, insertion, translocation & inversion.
4. Phylogenetic tree: diagram showing the inferred evolutionary relationships among species.
5. GUI: Graphical User Interface
6. Convergent evolution: Independent evolution of species that was not present in the shared common ancestor.
7. MoSCoW method: Prioritizing features by labeling them as: Must have, should have, could have, won't have.
8. Iteration (or sprint): A time restricted effort in which the team tries to complete a defined set of tasks, a term mostly used with the Scrum methodology.