PROGRAMMING LIFE CONTEXT
DELFT UNIVERSITY OF TECHNOLOGY

DNA is Not an Acronym

Product Planning

Joël Abrahams - joelabrahams - 4443268
Georgios Andreadis - gandreadis - 4462254
Casper Boone - cboone - 4482107
Niels de Bruin - ndebruin - 4375440
Felix Dekker - fwdekker - 4461002

May 2017

Contents

1 Introduction **2**

2 Product Roadmap **3**
2.1 Project Setup . . . 3
2.2 Running Prototype . . . 3
2.3 Data Scaling . . . 3
2.4 Layout and Navigation . . . 3
2.5 Visual Encodings . . . 3
2.6 Interval Annotation . . . 3
2.7 Semantic Zoom . . . 4
2.8 Eye-candy . . . 4

3 Product Backlog **5**
3.1 User stories of features . . . 5
3.1.1 Must Have . . . 5
3.1.2 Should Have . . . 5
3.1.3 Could Have . . . 6
3.1.4 Won't Have . . . 7
3.2 User stories of non functional requirements . . . 7
3.3 User stories of know-how acquisition . . . 7

4 Definition of Done **8**
4.1 Definition of Done for a Feature . . . 8
4.2 Definition of Done for a Sprint . . . 8

Glossary **9**

1 Introduction

We have been tasked with creating an application that can visualise large genome structures for use in biomedical research. Therefore, such an application must be well suited for use by researchers. This means the product has to be able to deal with large amounts of data and still be responsive. Lastly, it should give users the ability to quickly and efficiently draw meaningful conclusions from the data.

Creating such an application is no trivial task. Therefore, we need to clearly define what we hope to accomplish in the given time-frame. This means that we have to prioritize our efforts to produce the best product possible given the resources. To this end we have created a product planning detailing how we aim to create our product, called `Hygene`.

In section 2 we discuss the milestones of the project. In section 3 we elaborate on the milestones, prioritize what has to be done by using the MoSCoW method, discuss user stories of know-how acquisition, and discuss an initial release plan. Lastly, we define the definition of done in section 4.

2 Product Roadmap

Our client has defined a set of milestones for us. For each of them, we list a set of requested features and a due date. The epics of the product backlog are mainly formed by items of this roadmap.

2.1 Project Setup

Due on: May 5
Description: In this first milestone, we need to get everything up and running to facilitate later work on the product.

2.2 Running Prototype

Due on: May 12
Description: This milestone is the first step featuring actual functionality. By the due date there should be a program that reads in a file in the specified format and shows it in a basic User Interface (UI).

2.3 Data Scaling

Due on: May 19
Description: After having created a basic prototype, we now need to support larger, more realistic genomes. This means that we need to support center-point queries on very large graphs, which requires smart parsing and storage of the genome data.

2.4 Layout and Navigation

Due on: June 2
Description: This milestone focusses on layout of the graph on the screen. No edges may overlap any more (as may have happened before, in edge cases), and the UI should allow for a basic form of navigation through the graph.

2.5 Visual Encodings

Due on: June 9
Description: To give the user useful, clear information, we need to use visual cues to illustrate meta-data on nodes and edges. This includes colours, sizes, and borderline-weights.

2.6 Interval Annotation

Due on: June 16
Description: The user should be able to annotate paths in the graph and bookmark single nodes. These Quality-of-Life features enable the user to draw use-

ful conclusions about the data presented to them, while not getting lost in the forest of sequence nodes.

2.7 Semantic Zoom

Due on: June 23
Description: Showing the same data in top-level view as in low-level per-node view is simply not feasible: The data shown to the user needs to be curated. Based on the zooming level, a semantically useful level of detail should be displayed, letting the user focus on what matters.

2.8 Eye-candy

Due on: June 28
Description: This milestone is free to be filled in as the time for it approaches, but it generally includes any cosmetic features that can improve the workflow of users in the application even more. An example of this is a menu with meta-data that pops up while hovering over a node.

3 Product Backlog

3.1 User stories of features

To be able to meet the project goals, the applications needs to have some basic features. However, after this basic functionality is implemented, further extension of the application is possible. To keep track of priorities of features that make up the program, requirements are split up using the MoSCoW method.

The MoSCoW method is a prioritization technique which is commonly used in software development designed by Dai Clegg Clegg and Barker, 2004. It specifies four categories: Must haves, Should haves, Could haves and Would like but will not get. These categories prioritize the key features of the software product and place less priority on the extra features. In the remainder of this section we will list our tasks grouped by this categories.

3.1.1 Must Have

- The user shall be able to interactively explore a sequence graph representing the genome architecture of multiple strains.
- The user shall be able to load Graphical Fragment Assembly (GFA) files containing the genomes.
- The application shall be able to efficiently handle large genomes and keep information about them in an efficient data structure.
- The user shall be able to zoom in and out within the current (semantic) level of visualization of the genome.
- The user shall be able to semantically zoom in and out on the genome.
  - Collapse Single-Nucleotide Polymorphism (SNP) bubbles with different visual encoding.
  - Merge neighboring nodes.
  - Collapse complex bubbles.
- The application shall be able to identify different types of mutations (insertion, deletion, SNP) uniformly across the samples.
- The application shall be able to directly go to a certain position in the graph to identify mutations in the graph in the context of well-known reference genomes.
- The application shall be able to highlight paths of annotations and bookmarks.
- The user shall be able to visualize annotations from Browser Extensible Data (BED) annotation files.

3.1.2 Should Have

- The user shall be able to filter on types of mutation.
- The application shall have visual encodings for different classes of mutations.

- The application shall display a mini-heatmap of bubble occurrences and provide a 'sliding window' to quickly go to that part of the genome.
- The application shall display node/edge details on mouse over.
- The user shall be able to add bookmark, which are paths of bases which can be stored persistently on the user's computer.
  - the ability to create bookmarks.
  - the ability to remove bookmarks.
  - the ability to name bookmarks.
  - the ability to share bookmarks.
- The application shall have a data structure that can load parts of the genome into memory dynamically.

3.1.3 Could Have

- The user shall be able to see history of recently viewed genomes.
- The user shall be able to manually annotate genomes.
  - textboxes that can be placed around the genome.
  - a drawing tool to draw images or symbols such as arrows.
- The application shall use a cloud service to synchronise data.
  - to synchronise bookmarks and custom annotations.
  - to synchronise genome files.
  - a way to share files between user's clouds.
- The application shall have collaboration tools.
  - live sharing of bookmarks and annotations.
  - collaborative scrolling, where both users always see the same thing.
- The user shall be able to add tags to bookmarks to categorise them.
- The user shall be able to interactively edit genomes.
  - the ability to remove bases.
  - the ability to duplicate bases.
  - the ability to create new bases.
  - the ability to save the genome as a new file.
  - the ability to do this collaboratively.
- The application shall be able to calculate the similarities between different genome sequences.
- The application shall store the offset the user was at in the genome when they closed the application.
- The user shall be able to export the current view as an image or PDF.
  - a cropping tool.
  - the option to enable/disable annotations in the resulting file.
- The user shall be able to select different colour schemes.
  - a colour blind mode.
  - an interface theme.
  - a tutorial to introduce the user to the application.

3.1.4 Won't Have

- The application shall be a cloud application that runs entirely in the browser.
- The user shall be able to share bookmarks over social media.
- The application shall not use circles for visual encoding.

3.2 User stories of non functional requirements

Besides actual features, there are also requirements that are non functional (in the sense that they're not directly related to features of the application itself, but to the environment around it).

- The application shall be written in Java.
- The application shall be built using Gradle.
- The application shall support Windows 7 (and higher), macOS 10.11 (and higher), and Linux (Debian-based, Fedora, and CentOS).
- The application shall work independent of hardware architecture.
- The application shall be licensed under the Apache license.
- The application shall be developed using continuous integration as provided by Travis CI.
- The application shall be controlled under VCS as provided by Git.
  - The application shall be hosted on GitHub as a private repository.
  - The pull-based development model shall be used. Each pull request must be reviewed by at least two, but ideally more, team members.
- The application shall be developed using the Scrum methodology.
  - Sprints shall take one week.
  - GitHub Projects shall be used to manage user stories.
  - The complete development team shall meet on each friday afternoon. The meetings will take around 15 minutes.
  - The next sprint shall be planned during the meeting.
- All deadlines as described in the Context Guidelines shall be met.
- Communication shall occur over Slack for communication within the team. For communication with the TA, the course Slack team will be used.
- A user with domain experience shall be able to fully understand the application after 10 minutes of training.

3.3 User stories of know-how acquisition

Often times it can be difficult to know in advance what has to be built. In such cases it is common to do exploratory work, which often involves making a small prototypes and evaluating their performance and feasibility.

- As a developer, I want to prototype multiple alternatives for storing genomes in memory and evaluate which one is most suited for my application.
  - The found solution should be able to quickly and dynamically load data from disk.
  - The found solution should not required too much memory.
  - The found solution should allow quick lookups of data.

4 Definition of Done

In this section we define when we are done. This section is considered by the members to be a contract and should be referred to when code is peer reviewed and when the sprint comes to an end.

We give the definition of done for features and for sprints. Because a new release is published after every sprint, release requirements are listed under the section applying to sprints.

4.1 Definition of Done for a Feature

A feature is considered done when the following actions have been taken:

- The minimum production code for the feature has been written.
- Unit tests for the production code have been written such that the project's total branch coverage remains at an acceptable level.
- Integration tests for the production code have been written such that the peer reviewers of the production code have confidence in its proper functioning.
- The code has been peer reviewed and approved by at least two members that did not contribute to the feature.
- The code has been merged with the master branch in the Version Control System (VCS).
- The additional ”done” criteria specific to the feature have been satisfied.

4.2 Definition of Done for a Sprint

A sprint is considered done when the following actions have been taken:

- All features for the sprint are *done*.
- Slides and a demonstration for the weekly presentation have been prepared.
- Three screenshots best representing the current state have been made.
- The product backlog has been refined and a planning for the next sprint is has been made.
- The binary release has been made available and is functional on Windows, Linux, and macOS.

Additionally, a sprint retrospective should be produced after each sprint, but this is not part of the sprint itself.

Glossary

BED Browser Extensible Data. 5

center-point query a query on a graph, giving one node as central target and a radius around it, to establish which nodes should be visited. 3

genome the haploid set of chromosomes in a gamete or microorganism, or in each cell of a multicellular organism. 2

GFA Graphical Fragment Assembly. 5

MoSCoW MoSCoW method is a prioritization technique which is commonly used in software development. 5

SNP Single-Nucleotide Polymorphism. 5

UI User Interface. 3

VCS Version Control System. 8

References

Clegg, D. & Barker, R. (2004). *Case method fast-track: a rad approach*. Boston: Addison-Wesley.