Geex Final Report

Contextproject Programming Life - Group 2
TU Delft

Gerben Oolbekkink

4223896

Jasper Boot

1516272

Jasper Nieuwdorp

4215796

Jim Hommes

4306090

René Vennik

4102959

June 25, 2015

Contents

Abstract

Drug resistant bacteria are growing in numbers, and to create an understanding of how the drug resistance is developed software is needed. Multiple applications already exist to explore single genomes, but there is no tooling for visualising multiple genomes as of yet. To help research biologists from Broad Institute of MIT and Harvard and KwaZulu Natal Research Institute for Tuberculosis & HIV (KRITH) with this research the software contains semantic zooming, identification of mutations and integration with annotations from external sources. The software is meant to be used for comparing and highlighting genomes, viewing the phylogenetic tree, and displaying specific info of the genome set. The software is written in roughly 4,000 lines of Java 8 code, using dependencies such as JavaFX 8 and JGraphT.

The software is written to be user friendly: It does not require a manual, does not interfere with speed and data can not be represented ambiguously. Different points from interaction design were taken into account when developing the software, including testing with the customer. The outlook includes faster loading, searching nodes in graphs, user annotations and much more, but the possibilities for the future are endless.

1. Introduction

To translate genomes into a computer readable format lots of methods have been developed. *Genome sequencing* nowadays have low cost, in comparison to the past where computers had to work many hours to sequence all the data. The next step in this process is to decode or reverse engineer the genome. To do this, research biologists are interested in detecting variations in sequences between organisms. Detecting these variations can be done by constructing 'multiple sequence alignments' of complete genomes.

These multiple sequence alignments can be displayed in graphs. At the moment there are a lot of explorers available to study single sequences, but what these explorers lack is that they can not interpret these large graph structures. That is why Geex is created: Software that is able to visualise large multiple sequence alignments.

This document is the final report of the product *Geex*, that is developed for research biologists of Broad Institute of MIT and Harvard and KwaZulu Natal Research Institute for Tuberculosis & HIV (KRITH). In this document you will find all desired info of the software, including information on the functionalities and interaction design, the process of development and finally the evaluation. With that the goal of this document is to describe the product in detail and to inform on the road to the final product.

Geex is supposed to fill the gap in software that is able to compare "to interactively explore a sequence graph representing the genome architecture of multiple strains" (Abeel, 2015). The customer needs this software to analyse bacteria, such as Tubercolosis mutations. These bacteria are getting drug resistant (Mahr, 2013), and with the product research biologists are able to properly analyse a genome architecture in order to find out how this resistance develops.

To create a product that is suitable for this problem, end-user requirements were stated by the customer. These requirements included visualisation, identification of mutations by their variant and integration of annotations from sources like tbdb.org, TuberQ, tuberculist and PolyTB. The main point for visualization was semantic zooming "to enable useful visual interpretation at various zoom levels from whole-genome to individual mutations" (Abeel, 2015) and there should be visual encodings for different types of mutations.

Besides these requirements there are also some restrictions to the software, the most important being that the software should be able to handle hundreds of bacterial strains at the same time. With that comes handling thousands of edges, millions of bases per sequence and this should all work on a 32-bit system.

2. Software overview

This section will give an overview of the developed and implemented software product. The application can be divided in a welcome view, a phylogenetic view, a graph view, a view to mark strains and a help view. For the screenshots of the views, see section B of the Appendices.

2.1. Welcome view

The responsibility of the welcome view is to enable the user to select a workspace. This can be a new workspace or a previously selected workspace. Also, the welcome screen indicates to the user that the program has started.

2.2. Phylogenetic view

The phylogenetic shows the user the loaded phylogenetic tree. The main responsibility is to enable the user to select every possible subset of strains from the tree. A secondary responsibility is to show the user the phylogenetic data.

2.3. Graph view

The graph view enables the user to interact with the graph. The user can interact with the filters on the left side of the window. Filters can be added, removed and dragged-and-dropped. By clicking on the filters the graph can be displayed based on the stack of filters selected. At the bottom of the graph view there is a locator bar that indicates the position on the TKK_REF reference genome. Also, the locations of the known Drug Resistance Causing Mutations are shown on the locator.

The graph contains nodes and edges. There are 2 types of nodes: normal nodes and collapsed nodes. Normal nodes are the nodes that are loaded from the data. Collapsed nodes are combinations of normal nodes. Normal nodes are collapsed depending on the previously mentioned filters. The width of the edge indicates the amount of strains that pass through that edge.

It is also possible to mark strains in the graph view. This can be done by summoning the mark strains view. When strains are marked, nodes that are part of the marked strains are highlighted.

2.4. Mark strains view

The marked strains view is the filtered version of the phylogenetic view. It contains only the currently selected strains. By selecting any subset of strains, all the selected strains are instantly marked in the graph view.

2.5. Help view

The help view can quickly be called by pressing F1 or with the menu. It's an Ubuntu like help overlay. It contains all the shortcuts of program and a detailed legend.

3. Reflection on the product and process

This sections contains the reflection on the product and process from a software engineering perspective. The product is developed, using the Scrum Methodology. Within the team, a Scrum master (Jasper Nieuwdorp) and a Product owner (René Vennik) were chosen. The entire team was responsible for writing code.

3.1. Product

The product started small with an application in OpenGL. This was changed very quickly because coding on OpenGL would take up most of the time. The project started with parsing the graph. The parser hasn't changed much after that, but the graph models were re-factored about 3 times to keep up with the requirements.

Half-way through the project the decision was made to split up the project into multiple different maven modules. This has a lot of advantages. For example, there can't be any cycles between maven modules which enforces better programming.

3.2. Process

Since scrum was used as development methodology there was always a working build at master. With a total of over 1,500 commits and nearly 200 pull requests it was assured that the code base only changed bit by bit. This encouraged the team to react quickly on each other's changes in code, resulting in very few conflicts.

Each morning scrum meetings were held, this helped a lot with keeping track of the process. Each week a sprint plan was created to plan the next week, at the same time a sprint reflection was made to reflect on the previous week. Each week this process improved and estimations became better.

Most of the meetings were held at the home of one of the team members. Working together in the same room helped a lot with figuring out problems and coming up with ideas. It also made it possible to work at any time of the day together as a team.

3.3. Tooling

In order for the project to be manageable at a larger scale, several methods of static analysis were used.

3.3.1. Checkstyle

Checkstyle was used to to analyse the code for style issues like line length, unused imports, etc. For this reason a `checkstyle.xml` file with some custom rules was introduced.

- Design for extension
  This check was disabled, because enabling this check would result in sacrificing some unit testing. This is because final methods are impossible to mock. The code in the project is also not aimed towards direct extending.
- Line Length
  Currently the maximum line length is 120 characters instead of the default 80 characters, because all the team members' screens can show 120 characters or more. Using 80 characters enables blind programmers to read and contribute to the project. However, the decision was made to not support this.
- Private variables
  Most of the variables are package-local, because it is easier to test. Security isn't a very big issue, as the internet isn't accessed and the software doesn't store sensitive information. Using package-local variables makes testing a lot easier, because you can check information which only changes the inner structure of a class in a unit test.
- Hiding variables in Constructors
  Having the names of the fields in a class equal to the parameters in it's constructor is something can be very clear when used consistently. Hiding variables in functions is still disallowed.

- Magic numbers
  100 isn't a magic number in this project, because it is used for calculating percentage and is better to understand than a variable in this place.

3.3.2. Findbugs and PMD

The standard configuration for Findbugs and PMD was used. Pull requests with any warnings were not allowed to be merged before these errors were fixed. Octopull was also setup in order for pull request to include static analysis.

At the end of the project several modules were introduced, this caused some regression in the static analysis. This was because Findbugs doesn't support aggregate reports and couldn't create a single report for the entire project.

3.3.3. SonarQube

At the end of the project SonarQube was introduced. SonarQube provides an all in one solution for static analysis. It helped with fixing some Java 8 and dependency issues the other tools didn't detect.

3.4. Feedback

Most of the feedback was received during the meetings with the teachers and student assistants each week. There were weeks where the meetings had a very short time between them, this was not very useful.

The feedback on the software engineering part was quite poor, but luckily the last part of the project a new student assistant took over and gave some very useful feedback.

4. Description of the developed functionalities

There are a few key functionalities in the implementation that really make it stand out and give it an edge in this field. Over the next few paragraphs they will be listed, explained and some context will be provided in why this functionality is useful for the end-user.

4.1. Comparing strands

Geex offers the ability to load reasonably large sets of sequenced DNA strands and compare them to each other. When all the data is loaded a graph will be shown containing all the information of the strands. In current solutions to visualise genomes it is common to view only one genome at a time. When researching drug resistance or other mutations it's a lot more interesting to look at the relations of multiple strands to understand what's happening in the genome. (see appendix B figure 3)

4.2. Phylogenetic tree

The core functionality of the project is to navigate through sets of genomes, rather than look at only one. To show the relation between the loaded strands Geex uses a phylogenetic tree to put

the strands in context with each other. It allows the user to select common ancestors of strands which is a very useful feature to research parallel convergent evolution. (see appendix B figure 2)

4.3. Strand Highlighting

Another way the user might look at the data is looking at the relation of one or more strands to the rest of the strands. In order to achieve this there's a function to highlight strands in the graph, so that the user can follow them throughout the graph. This helps identifying parts of the graph that might be interesting when researching a specific strand or mutation. (see appendix B figure 4)

4.4. Annotations

To incorporate earlier research in the visualisation Geex offers the ability to load annotations. Annotations can be used to give information about whether certain parts of the genome are coding or where mutations are that are already identified to cause resistance to certain drugs. The information from the annotations is used at multiple levels in the graph, allowing the user to see which parts of the graph are interesting. For example they are visible in the graph, indicated by small circles under the nodes. Most information about the annotation is available when looking at the node itself.

4.5. Semantic Zooming

Due to the complexity of the data and the medium on which it is displayed semantic zooming is required to avoid unclear representation of the data. The user can add and order filters to create zoom levels. When the criteria of the filter are met nodes below are collapsed into one node. This collapsed node now contains a sub-graph of the elements that are collapsed. It is possible to click through the (sub-)graph to the next zoom level. Another way the user can zoom is by enabling or disabling filters. Combining filters can help the user by only showing what is relevant for the user.

4.6. Filtering

In order to keep the large graphs manageable and to keep the visualisation clear, there's a feature that uses filtering to only keep the information that is of use to the user at that time. When a filter is activated, all nodes of the graph that meet the criteria of the filter are either collapsed (joined together in 1 node) or removed from the graph. When a user is looking at data using Geex, not all the data the program has to offer is relevant to the user. Filters help by giving the user control over the graph for their specific research. (see appendix B figure 3)

4.7. Weighted edges

When exploring the data it is interesting to see what strands differ from the general consensus. If all of the strands that are resistant to a certain drug deviate from the rest of the strands at one place in the graph it is very likely that the mutation at that place plays a role in causing this resistance. By indicating which edges are followed by most of the strands at any point in the graph, it is easily identifiable which paths are the most common and which parts are rarer. Combined with the strand highlighting mentioned earlier (4.3) this gives the user a powerful tool to find interesting parts of the genome. (see appendix B figure 3)

4.8. Base percentages

Distribution of the different bases can be used to quickly identify nodes that have similarities or are vastly different. It can give the user a sense of the complexity of a certain node. In Geex the percentages in which the bases occur are displayed on the nodes in the graph. This way the information can be seen without focusing on just one node without cluttering the graph. This feature might help the user to discover mutations in the graph such as translocations and inversions. (see appendix B figure 3)

4.9. User Interface

The user interface was kept as clear as possible. For more information about this process please refer to 5. At every point in the program the menu bar at the top is visible. This is the standard menu bar offered by the OS of the user. Additional information will be shown in an overlay and there can be only one overlay at a time. For semantic zooming there's a sidebar with a list of filters. For screenshots please refer to appendix B

5. Interaction design

To ensure that the end-user can use the product effectively and with ease without sacrificing the functionality of the program, the team incorporated the User Centred Design (UCD) process during the development of the product. This means that the product is purely designed for the end user instead of for the computer or the developers. Using UCD fits well in the agile methodology that was used in this project, since UCD is iterative. It also states that the developers and in a lot of cases the customer (where the customer is not the end-user) does not know what will work for the user. To find out what would be the best design for the program a few established techniques were used. In this section the used techniques are described.

5.1. Cultural probe, Persona and Task Analysis

Getting a clear idea of the end-user early on in the project a certain knowledge about the context and environment in which the program would be required. Inspiration came from lectures about the context and by reading literature and looking up current solutions in the prospected environment (where applicable). Conversations with the customer also allowed envisioning the field in which the program will be used. After this phase, called the *cultural probe*, there was enough information to construct a persona. A fictional archetype used to model the end-user so that developers can reason about the desires and preferences of said persona. For this project the persona was as follows:

Francis is educated at a master's level or higher in the field of genomics or genetics and he has basic knowledge about computers, but has no prior knowledge about any particular programming language or specific specialised applications. He knows how to use computers but is not specifically interested in how they work. However, assumed is that Francis has the ability to quickly learn to use specialised programs, by trying out intuitive features that the program provides. Francis is most likely in the early stages of a research or wants to use the program to validate assumptions made earlier. Francis is fairly patient and calm and cares more

about correctness of data than the allure, as long as it doesn't sacrifices clearness. Francis has a desire constantly keep learning. Francis speaks English at an academic/proficient level.

Because the product makes it possible for existing data to be used in a new and different way it was not possible to create an accurate task analysis at the start of the project. The tasks would flow from the form of the project which gave the developers a lot of freedom in shaping the possibilities of the program. In lieu of an accurate Task Analysis a few dogmas were created and when a design question arose the dogmas were applied to our persona (Francis).

1. Screen-space must be used efficiently.
2. The program must not require a manual.
3. Design must not interfere with speed.
4. Data can not be represented ambiguously.

An example of this process can be found in how the phylogenetic trees should be visualised in the software. There are 3 main schools of thought: In a circle as a sunburst or as rectangular tree with straight edges and sharp corner. (See appendix D) When looking at the dogmas the rectangular representation already comes out ahead. The screen is also rectangular so a circular representation would mean empty space in the corners; and this conflicts with the first dogma. A circular representation also decreases the readability because the text in the larger part of the circle will be at an angle that is difficult to read. So controls to rotate the tree should be available for the user. This makes the software more difficult to learn and quite possibly requiring a manual (dogma 2) and slows down looking for certain strands (dogma 3). So while the circular representation might be thought off as more visually appealing, Francis would prefer the rectangular representation.

5.2. Contextual Inquiry

The customer does match the description of the persona, so during some of the weekly meetings the customer was given control over the program in the current state (a high fidelity prototype) and the behaviour was observed. Special attention was given to when the user stopped to look for something, did something that was not his intention and asked a question. The customer was encouraged to vocalise freely while using the program and express any ideas or concerns. The customer was not afraid to ask for changes to the product and often came up with interesting perspectives that the developers did not think of yet. This lead to the decision that a low fidelity prototype would not be beneficial at this stage.

5.3. Product Design

A few concrete design choices were made that would reappear in the entire product. This subsection elaborates on the most important ones.

5.3.1. Colours

When choosing colours it was taken into account that contrast was important to see clearly. This lead to the important choice between a bright colour scheme or a dark one. Since there wouldn't be a lot of reading in the program, the choice was made in favour of the dark colour scheme (Bauer et al., 1980).

It was desirable that colourblind people could also make clear distinctions between the colours, printing the information was not an feature so the used colour scheme was not necessarily colourblind/photocopy safe, but was colourblind safe. It was also made sure to use qualitative colour schemes where applicable, and transparency to indicate sequences (Brewer, n.d.).

When attention was required from the user we would always use bright colours, in contrast to the milder colours used to visualise the data. Bright red meant something negative and blue would indicate action. The importance of attention would follow from the size of the object.

5.3.2. Discoverability

When menus and information had to be shown the question was always: does this need to be visible right now. If it wasn't it would be hidden at this stage, only to be accessed through the main menu bar and/or a keyboard shortcut. In the occasion would occur that the user would want to learn more about the product (for example shortcuts) there would be only one place for help so that the user would not be lost. The historically typical key for help is F1 which was also how the help overview was triggered.

In the program the user was actively encouraged to use shortcuts by displaying them every time he used his mouse to use a feature. This way the user automatically sees often used shortcuts, while not being bothered by them, appearing and demanding attention when he has no desire to use shortcuts.

[reference slides from beginning]

5.4. Usability analysis

The usability analysis is based on five tasks that user had to perform. The metrics are the amount of clicks, the amount of key strokes and the degree of help that was given. The data is available in Appendix C.

5.4.1. Findings

In general the findings were that the program is highly scientific and hard to explain to regular users. Simple tasks can become quite hard if the user has almost no context. Most of the users did not use the menu (and it's shortcuts) and were looking for buttons in the view.

Most users had no clue at what they were looking at and the concept of a graph was not even clear to everyone.

5.4.2. Future improvements

During the analysis it was clear that there were some usability issues. Further improvements of the application should therefore include the following:

- The menu item that opens the view to mark strains should be renamed to something that describes what it does.
- More buttons should be added to improve navigability inside application with the use of a mouse.

6. Evaluation

The product, given as it is, has a lot of responsibilities, served by specific functionalities that have been tweaked over the last two months. This section elaborates on how and why these functionalities were established by the team. First, an evaluation of the functional modules are given, followed by an evaluation of the product in its entirety, together with a failure analysis.

6.1. Modules

Roughly after five weeks the modules were introduced to make clear use of responsibilities and dependencies. The following modules are used in the final product.

- Loading graph, phylogenetic, annotation and mutation data
- Phylogenetic tree
- Genome graph
- Mini map
- Filtering
  - By phylogeny
  - By mutation type

6.2. Product

During the first iteration some rough specifications were specified by the customer and team to form a base for the program. Later on it got much clearer what the expectations of the application were. The most important features were listed in a MoSCoW structure and this subsection evaluates the absence or implementation of the each feature.

6.2.1. Must haves

Nearly all must haves were implemented the way they were meant to be. The most basic – yet most important – features that have been implemented, were the loading of the available data (graph, phylogenetic tree and annotations). Also the use of filters, mini-map and phylogenetic tree were implemented in a way that was easy to use and clear to the user.

The annotations (coding sequence annotations and drug resistance causing mutations) were implemented in a way that they were discoverable, but lacked a view in which you could quickly identify where the annotations were at. Also the implementation of semantic zooming might be different from what one might expect; instead of zooming with mouse scrolling or a scroll button, zooming is done while stacking multiple filters. This gives the user more freedom in what he wants to see and how he wants to see it.

Zooming out the furthest is, in fact, done using all possible filters that the application provides. This results in only a few nodes that could fit on one screen, depending on the screen resolution. A better way could have been an entirely different view, which might not even have been a graph. It could be some kind of view that shows the strains in its entirety that reveals in some way the interesting parts of the strains.

6.2.2. Nice to haves

First of all, the some important features were deemed very important and were implemented from the start. For example, the applications opens in full-screen mode to view as much information as possible and the views have a somewhat 'modular' layout to divide the responsibilities across the screen. Also the use of a so-called workspace was introduced early on. This resulted in a very user friendly interface to quickly select a folder which contains all the data files.

Other features were implemented later on. One of these features was the magnification function of the mouse while hovering over the nodes in the genome graph. Looking back, this might have been a less important feature and the time invested in this could have been spent on something more important.

Lastly some features were left out because they could be easily implemented later on, if we decide to continue further development after the contextproject's life cycle. Features, such as exporting to a HTML5, JavaScript and CSS format.

6.3. Failure analysis

Most of the features were implemented and they were implemented well. The only thing that stood out that was the loading time and 'lag' while using huge sets. Loading the 671 strains set won't work, because it uses to much memory. We could have prevented this if we had used a graph database like Neo4j instead of JGraphT. This would have made the load time a little bit slower, but overall the application could have handled bigger data sets.

7. Outlook

While the application already has a substantial amount of features that distinguishes the application to its competition, there is always room for improvement. Therefore, should the development of the application be continued, it is only logical to think about possible improvements and the strategy on how to implement these improvements. This section highlights the main features that could add great value to this application.

7.1. Faster loading

The application is quite fast and already uses preprocessing, caching and on-the-fly loading techniques. However, especially big graphs, take up some time to load and make the application feel less fluent. More techniques could therefore be introduced in a way that the user has to wait as little as possible and processed data could, perhaps, be stored for interaction on the next time that the application is used.

7.2. Searching in graphs

The graphs can contain many and very long strands. It could therefore be a great relief to find an integrated search bar that lets the user quickly jump to certain known annotations or base sequences. This would improve the usability of the program and saves the user from searching and scrolling to the desired location in the graph. This could easily be done with known annotations, but to match base sequences, some preprocessing and perhaps indexing would come in hand to be able to show the user the results in a reasonable time for huge graphs.

7.3. User annotations

Sometimes you would want to create your own comment or annotation on a certain base sequence in the graph. This makes it easier for the user to identify those sequences and perhaps even add or alter information about these sequences. The application could be extended with the ability to create, update and delete annotations. Also, this is only useful if the user can reuse these annotations. That's where an export functionality comes in handy, so that the user can use these annotations the next time he uses the application.

7.4. Crossing edges optimisation

Despite the fact that the graph is shown in a nice way, most of the times, it can get messy in certain places where a lot of strands take a different path. It could be made easier on the eyes if as much crossing edges could be filtered out as possible. For this an algorithm could be used to reposition the nodes that cause the crossing edges. Nevertheless, there could also be situations in which it is not possible to have zero crossing edges because of the way that nodes are connected.

7.5. Exporting data

Exporting data was specified as a nice to have by the customer, this could also be implemented in the software. The user could for instance export the current graph as a specific graph file. This data could then be used to further analyse the structure. Saving the current state of the software could also be an export option. This way, a user can save the state and return to it at a later moment.

7.6. In-graph annotations

The textual annotations of the strains are currently only visible when a certain node is selected that has an annotation. For researching purposes it would be nice to see the annotations in-graph at a glance. This way the user can quickly identify its relative position in the genome graph.

References

Bauer, et al. (1980). Improving the legibility of visual display units through contrast reversal.

Brewer, C. (n.d.). *Colorbrewer: color advice for maps.* http://colorbrewer2.org. (Accessed: 2015-06-25)

Maisch, M. W. (n.d.). Phylogeny, systematics, and origin of the ichthyosauria–the state of the art.

Pearse, W. (n.d.). *Will's eco-evo r(d).* https://willeerd.wordpress.com/tag/phylogeny/. (Accessed: 2015-05-28)

ram.dsramesh. (n.d.). *To create protovis sunburst charts : Python script to create dataset in json format (or) parent - child json.* https://www.biostars.org/p/68102/. (Accessed: 2015-05-28)

Appendices

A. Project dependencies

A.1. Production

- JGraphT
- TreeJuxtaposer (cjb/libnewicktree)
- SQLJet
- Apache Commons:
  - Apache Commons Lang
  - Apache Commons Collections

A.2. Development

- Maven
- Testing:
  - JUnit
  - Mockito
  - JavaFX JUnit-4 Testrunner
- PlantUML
- Code analysing tools:
  - CheckStyle
  - FindBugs
  - PDM
  - CPD
  - Octopull
  - SonarQube

B. Screenshots of Geex

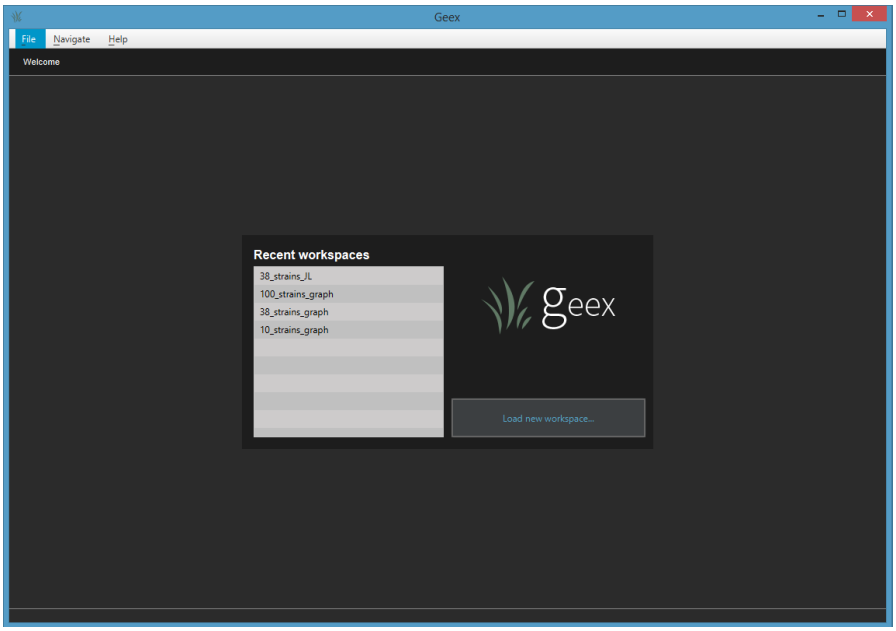

Figure 1: Welcome view of Geex

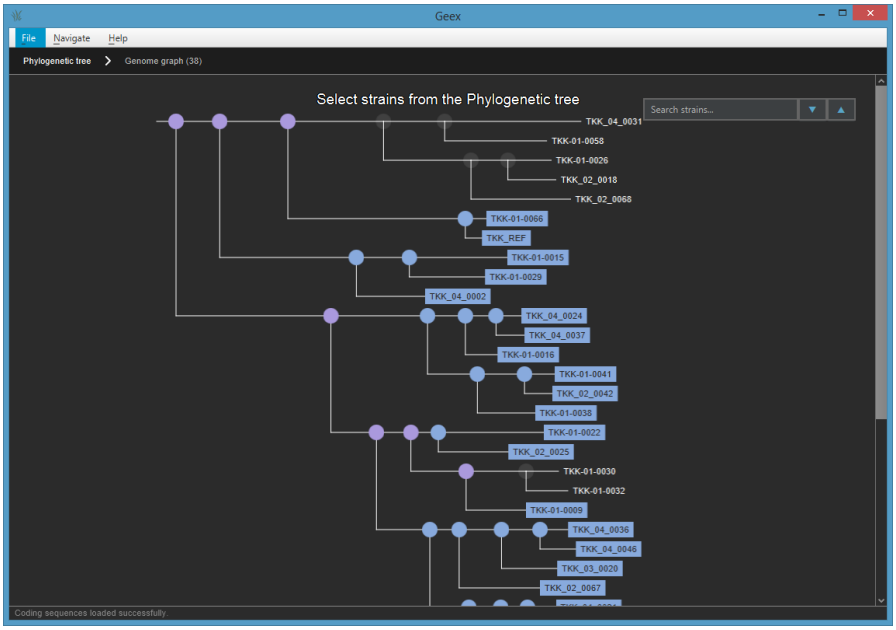

Figure 2: Phylogenetic view of Geex

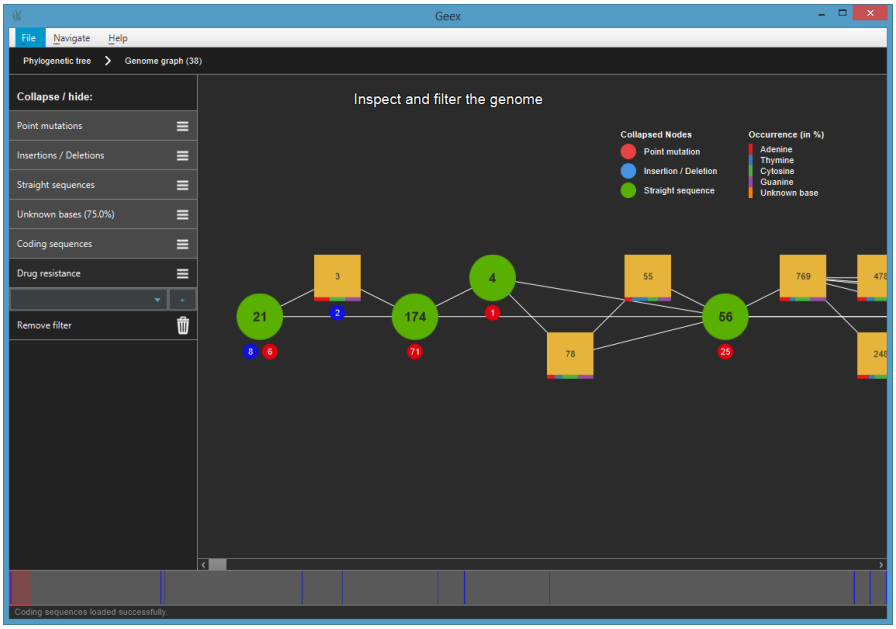

Figure 3: Graph view of Geex

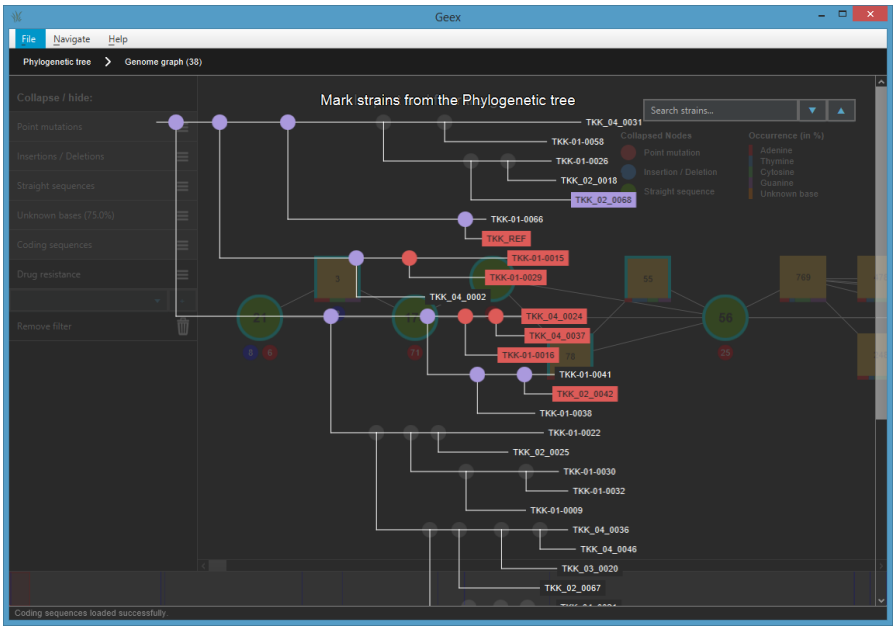

Figure 4: Mark strains view of Geex

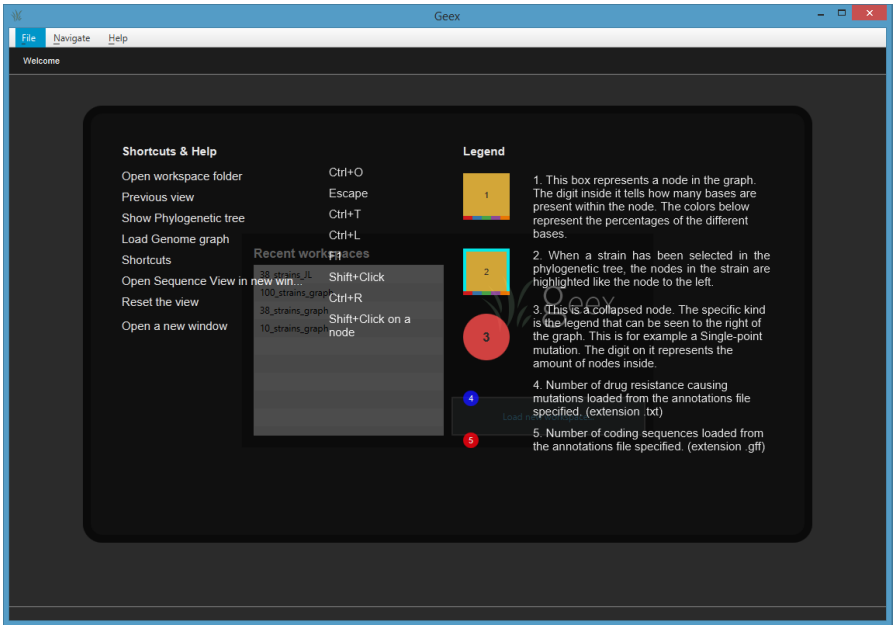

Figure 5: Help view of Geex

C. Usability analysis data

Table 1: Task 1. Open a workspace, and select all strains, except for one and load the graph.

Person	Age	Education	Gender	Clicks	Keys	Help
Nieuwdorp A	23	Master	F	8	0	pointer
Nieuwdorp B	54	HBO	M	10	2	walk through
Nieuwdorp C	19	Highschool	F	8	0	pointer
Oolbekkink A	48	Master	M	10	0	pointer
Oolbekkink B	20	Bachelor	F	5	0	-
Hommes A	20	Bachelor	F	4	0	-
Hommes B	49	Bachelor	M	3	0	-
Vennik A	22	HBO	M	7	0	pointer
Vennik B	23	Master	M	4	0	-
Boot A	25	HBO	F	3	0	hint
Boot B	27	Bachelor	M	3	0	-
Boot C	25	Bachelor	M	1	0	-

Table 2: Task 2. Open the overview of shortcuts.

Person	Age	Education	Gender	Clicks	Keys	Help
Nieuwdorp A	23	Master	F	15	2	-
Nieuwdorp B	54	HBO	M	5	1	hint
Nieuwdorp C	19	Highschool	F	0	2	none
Oolbekkink A	48	Master	M	10	0	pointer
Oolbekkink B	20	Bachelor	F	5	0	-
Hommes A	20	Bachelor	F	4	0	-
Hommes B	49	Bachelor	M	3	0	-
Vennik A	22	HBO	M	7	0	pointer
Vennik B	23	Master	M	4	0	-
Boot A	25	HBO	F	15	5	hint
Boot B	27	Bachelor	M	10	5	-
Boot C	25	Bachelor	M	15	0	-

Table 3: Task 3. Zoom in on the graph until the lowest level.

Person	Age	Education	Gender	Clicks	Keys	Help
Nieuwdorp A	23	Master	F	7	0	pointer
Nieuwdorp B	54	HBO	M	8	0	walk through
Nieuwdorp C	19	Highschool	F	8	1	pointer
Oolbekkink A	48	Master	M	10	0	pointer
Oolbekkink B	20	Bachelor	F	5	0	-
Hommes A	20	Bachelor	F	4	0	-
Hommes B	49	Bachelor	M	3	0	-
Vennik A	22	HBO	M	7	0	pointer
Vennik B	23	Master	M	4	0	-
Boot A	25	HBO	F	20	5	hint
Boot B	27	Bachelor	M	10	5	-
Boot C	25	Bachelor	M	20	0	-

Table 4: Task 4. Mark the strand TKK-Ref in the current graph.

Person	Age	Education	Gender	Clicks	Keys	Help
Nieuwdorp A	23	Master	F	3	2	pointer
Nieuwdorp B	54	HBO	M	4	3	pointer
Nieuwdorp C	19	Highschool	F	3	2	pointer
Oolbekkink A	48	Master	M	10	0	pointer
Oolbekkink B	20	Bachelor	F	5	0	-
Hommes A	20	Bachelor	F	4	0	-
Hommes B	49	Bachelor	M	3	0	-
Vennik A	22	HBO	M	7	0	pointer
Vennik B	23	Master	M	4	0	-
Boot A	25	HBO	F	20	5	hint
Boot B	27	Bachelor	M	10	5	-
Boot C	25	Bachelor	M	20	0	-

Table 5: Task 5. Open a different workspace.

Person	Age	Eductation	Gender	Clicks	Keys	Help
Nieuwdorp A	23	Master	F	4	1	pointer
Nieuwdorp B	54	HBO	M	4	3	pointer
Nieuwdorp C	19	Highschool	F	5	3	hint
Oolbekkink A	48	Master	M	3	0	-
Oolbekkink B	20	Bachelor	F	3	0	-
Hommes A	20	Bachelor	F	3	0	-
Hommes B	49	Bachelor	M	2	0	-
Vennik A	22	HBO	M	3	0	-
Vennik B	23	Master	M	2	0	-
Boot A	25	HBO	F	4	0	-
Boot B	27	Bachelor	M	4	0	-
Boot C	25	Bachelor	M	2	0	-

D. Phylogenetic tree representations

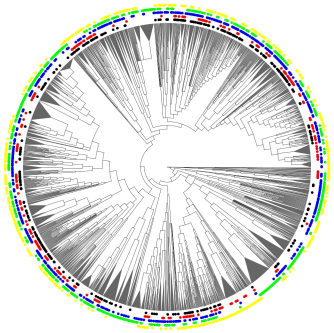

Figure 6: Circular representation of a phylogenetic tree
(Pearse, n.d.)

Figure 7: Sunburst representation of a phylogenetic tree
(ram.dsramesh, n.d.)

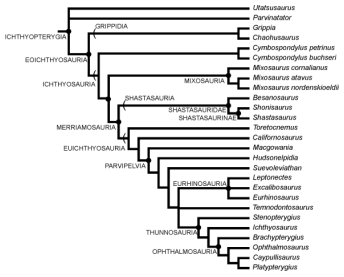

Figure 8: Square representation of a phylogenetic tree
(Maisch, n.d.)

E. Sprint plans

The following pages contain the sprint plans.

Sprint plan # 1

Contextproject: Programming life
Group: 2

User Story	Task	Task Assigned To	Estimated Effort per Task (1-5)
As the scrummaster I want to have a contract describing the rules obeyed by the team members	1. Write a collaboration contract 2. Agree to collaboration contract	1. Nieuwdorp 2. All	1. 3 2. 2
As a user I want to be able to see a phylogenetic tree from the genomes.	1. Parse the tree file into JGraphT 2. Display the simple version of the tree graph	1. Vennik 2. Vennik	1. 2 2. 4
As a user I want to be able to read the graph files.	1. Parse graph files into JGraphT 2. Display a simple version of the graph	1. Vennik 2. Vennik	1. 2 2. 5
As a programmer I want to set up a specific programming environment with all the desired tools.	1. Set up Travis 2. Set up Octopull 3. Set up Checkstyle 4. Set up Maven 5. Set up Cobertura 6. Set up PMD 7. Define a document flow	1. Oolbekkink & Vennik 2. Hommes 3. Oolbekkink 4. Oolbekkink 5. Oolbekkink 6. Oolbekkink 7. Nieuwdorp	1. 2 2. 3 3. 1 4. 1 5. 1 6. 1 7. 2
As a product owner I want to have a backlog with all potential features for the application.	1. Add features to the issue tracker on GitHub 2. Judge and prioritize issues from GitHub	1. All 2. Vennik	1. 3 2. 2

Sprint plan # 2

Contextproject: Programming life
Group: 2

User Story	Task	Task Assigned To	Estimated Effort per Task (1-5)
As a team we want to deliver our deliverables.	1. deliver sprintplan3.pdf 2. deliver sprintreflection2.pdf 3. deliver productplanning.pdf 4. deliver final productvision.pdf	1. Boot 2. Boot 3. Nieuwdorp 4. Boot	1. 1 2. 1 3. 1 4. 1
As a team we want to create and maintain our documents.	1. Review and redact the product vision based on received feedback. 2. Review and redact the arch. design based on received feedback. 3. Create a product planning	1. Nieuwdorp 2. Hommes 3. Nieuwdorp	1. 4 2. 3 3. 3
As a user I want to be able to see the genome graph with edges between the nodes.	1. Draw edges between the nodes.	1. Vennik	1. 3
As a user I want to have a clear & efficient view of the genome graph	1. Use an algorithm to filter out the crossed lines. 2. Show % of ATCG per node on high-level view 3. Enlarge the scrollbar for bigger graphs 4. Research if a dropdownmenu is better than the menu used for nodes and edges loading 5. If so, implement a dropdown menu	1. Vennik 2. Nieuwdorp 3. Hommes 4. Hommes 5. Hommes	1. 5 2. 4 3. 3 4. 2 5. 3
As a user I want to see the DNA-sequence that belongs to each node.	1. Display DNA-sequence on node.	1. Vennik	1. 3

Sprint plan # 3

Contextproject: Programming life
Group: 2

User Story	Task	Task Assigned To	Estimated Effort per Task (1-5)
As a user I want to see a phylogenetic tree minimap of the active genomes in the graph.	1. Create a view. 2. Display relative tree with lowest common ancestor as root. 3. Highlight the path between all active genomes in the tree.	1. Boot 2. Boot 3. Boot	1. 1 2. 3 3. 2
As a team we want to deliver our deliverables.	1. Deliver Sprintplan4.pdf. 2. Deliver Sprintreflection3.pdf. 3. Deliver Productplanning.pdf (final).	1. Nieuwdorp 2. Nieuwdorp 3. Nieuwdorp	1. 1 2. 2 3. 2
As a user I want to be able to see the difference between 2 paths in the graph.	1. Create selector for 2 paths. 2. Create a difference view.	1. Vennik 2. Vennik	1. 3 2. 3
As a user I want to have a clear & efficient view of the genome graph.	1. Show % of ATCG per node on the node. 2. Use an algorithm to filter out the crossed lines (from #2).	1. Nieuwdorp 2. Vennik	1. 4 2. 5
As a user I want to see what parts of the genome are important.	1. Detect important parts (clarification during meeting) 2. Display important parts in the view.	1. Oolbekkink 2. Oolbekkink	1. 4 2. 4
As a user I want to see which mutations are present	1.Explore how to identify different mutations. 2. Display mutations in the graph.	1. Hommes 2. Hommes	1. 3 2. 4
As a user I want our documentation to be up to date	1. Maintain ArchDesign.pdf 2. Finalize Productplanning.pdf based on feedback 3. Update UML diagrams	1. Nieuwdorp 2. Hommes 3. Oolbekkink	1. 3 2. 2 3.

Sprint plan # 4

Contextproject: Programming life
Group: 2

User Story	Task	Task Assigned To	Estimated Effort per Task (1-5)
As a programmer I want our documentation to be up to date so I can see the large lines of the project.	1. Update UML diagrams.	1. Oolbekkink	1. 2
As a team we want to deliver our deliverables.	1. Deliver Sprintplan5.pdf. 2. Deliver Sprintreflection4.pdf.	1. Nieuwdorp 2. Nieuwdorp	1. 1 2. 1
As a user I want to see which mutations are present.	1. Display mutations in the graph.	1. Hommes	1. 5
As a user I want to see the genome annotated with information from current sources.	1. Select available sources to use. 2. Transform our current coordinate system into other available coordinate systems so we can find the correct available resources.	1. Nieuwdorp 2. Nieuwdorp	1. 2 2. 3
As a user I want to be able to reduce the amount of information (semantic zooming).	1. Select information that needs to be reduced.	1. All	1. 3
As a user I want see what genomes I'm currently looking at without loosing my current view.	1. Make it possible to switch back to the phylogenetic tree and see your old selection (see user story below). 2. Remember old views until you make a view at the same level. 3. Make it possible to go back to the old views.	1. Vennik 2. Vennik 3. Vennik	1. 2 2. 2 3. 2

Sprint plan # 5

Contextproject: Programming life
Group: 2

User Story	Task	Task Assigned To	Estimated Effort per Task (1-5)
As a programmer I want our documentation to be up to date so I can see the large lines of the project.	1. Update UML diagrams.	1. Oolbekkink	1. 2
As a team we want to deliver our deliverables.	1. Deliver Sprintplan6.pdf. 2. Deliver Sprintreflection5.pdf.	1. Nieuwdorp 2. Nieuwdorp	1. 1 2. 1
As a user I want to have a clear & efficient view of the genome graph.	1. Use an algorithm to filter out the crossed lines (from #2).	1. Oolbekkink	1. 5
As a user I want to be able to reduce the amount of information (semantic zooming).	1. Create a form in the old control area with a (live) reload button. 2. Filter the information from the graph.	2. Vennik 3. Vennik	1. 4 2. 4
As a user I want to load the genome graph using menu items.	1. Make menu items accessible from other controllers. 2. Create a menu item to load the current strains, selected in the tree. 3. Create a shortcut	1. Boot 2. Boot 3. Boot	1. 4 2. 2 3. 1
As a user I don't want to wait for the same graph to load, when I try to view the same selection in the phylogenetic tree.	1. Store last selection of the tree 2. Compare the new selection of the tree to the old one 3. Go back to the old view if the selection hasn't changed	1. Boot 2. Boot 3. Boot	1. 3 2. 2 3. 1

Sprint plan # 6

Contextproject: Programming life
Group: 2

User Story	Task	Task Assigned To	Estimated Effort per Task (1-5)
As a programmer I want our documentation to be up to date so I can see the large lines of the project.	1. Update UML diagrams.	1. Oolbekkink	1. 2
As a team we want to deliver our deliverables.	1. Deliver Sprintplan7.pdf. 2. Deliver Sprintreflection6.pdf.	1. Nieuwdorp 2. Nieuwdorp	1. 1 2. 1
As a user I want to know at what position the nodes are that I'm viewing.	1. Think of a way to display it. 2. Implement the visualization of the user's position in the graph	1. All 2. Boot	1. 3 2. 4
As a user I want to be able to collapse nodes that are of less importance.	1. Make nodes collapsable	1. Vennik	1. 2
As a user I want to see the complete strains on the initial screen	1. Think of a way to fit all nodes on the first screen. (what information to display)	1. All	1. 3
As a user I want to be able to highlight certain nodes	1. Make nodes selectable and able to be highlighted. 2. Enlarge nodes that are near the mouse.	1. Nieuwdorp 2. Boot	1. 4 2. 4
As a user I want to see information related to the genome from other sources.	1. Transform our current coordinate system into other available coordinate systems so we can find the correct available resources. (from #4). 2. Display Mutations in the graph. (from #4)	1. Nieuwdorp 2. Hommes 3. Oolbekkink	1. 4 2. 5 3. 5

Sprint plan # 7

Contextproject: Programming life
Group: 2

User Story	Task	Task Assigned To	Est. Effort (1-5)
As a programmer I want our documentation to be up to date so I can see the large lines of the project.	1. Update UML diagrams.	1. Oolbekkink	1. 2
As a team we want to deliver our deliverables.	1. Deliver Sprintplan8.pdf. 2. Deliver Sprintreflection7.pdf.	1. Nieuwdorp 2. Nieuwdorp	1. 1 2. 1
As a user I want to highlight paths in the graph of certain strands.	1. Retrieve subgraph from existing Newick tree. 2. Select the strands that you want to highlight 3. Highlight the paths of the selected strands.	1. Boot 2. Boot 3. Nieuwdorp	1. 3 2. 2 3. 3
As a user I want to see the annotations & known mutations	1. Connect the annotations & known mutations to nodes in the graph 2. Display the loaded annotations in the graph	1. Nieuwdorp 2. Boot	1. 4 2. 4
As a user I want to quickly locate current view in the overall graph.	1. Draw a position bar at the bottom. 2. Show mutations in the position bar.	1. Vennik 2. Hommes	1. 4 2. 3
As a user I want to be able to see the entire graph at once and use semantic zooming, based on ‘interestingness’.	1. Create more filters. 2. Create filters based on mutations. 3. Create filters on base lengths.	1. Vennik 2. Vennik 3. Oolbekkink	1. 5 1. 4 1. 4
As a user I want to reset my graph view to the beginning.	1. Create a reset view button in the menu. 2. Reset the view on button click.	1. Hommes 2. Hommes	1. 1 2. 2

Sprint plan # 8

Contextproject: Programming life
Group: 2

User Story	Task	Task Assigned To	Est. Effort (1-5)
As a user I want to be able to see the entire graph at once and use semantic zooming, based on ‘interestingness’.	1. Create a zoom level that show the whole graph at once. 2. Come up with a way to calculate an interestingness factor for the graph. 3. Collapse nodes based on the interestingness factor and a threshold set by the zoom level.	1. Vennik 2. Oolbekkink 3. Vennik	1. 4 2. 2 3. 4
As a user I want to see the known mutations that cause resistance	1. Connect the known mutations to the graph. 2. Display the loaded annotations in the graph. 3. Display the annotations in the locator (minimap bar).	1. Boot 2. Boot 3. Boot	1. 2 2. 3 3. 4
As a team we want to be at the right level for our final report.	1. See which parts are missing for the final report. 2. Review the current final report. 3. Create a tasklist per team member for the final report.	1. Nieuwdorp 2. Nieuwdorp 3. Nieuwdorp	1. 1 1. 2 1. 2
As a team we want to deliver our deliverables.	1. Deliver Sprintplan9.pdf. 2. Deliver Sprintreflection8.pdf.	1. Nieuwdorp 2. Nieuwdorp	1. 1 2. 1
As a programmer I want our documentation to be up to date so I can see the large lines of the project.	1. Update UML diagrams. 2. Update the Architecture Design	1. Oolbekkink 2. Nieuwdorp	1. 2 2. 2
As a user I want to be able to save my configuration for the program.	1. Create a .ini where all settings are saved	1. Hommes	1. 4

F. Sprint reflections

The following pages containt the sprint reflections.

Reflection on Iteration #1

Context project: Programming life
Group: 2

User Story #	Task #	Task Assigned To	Estimated Effort	Actual Effort	Done (y/n)	Notes
As the scrummaster I want to have a contract describing the rules obeyed by the team members.	1. Write a collaboration contract 2. Agree to collaboration contract	1. Nieuwdorp 2. All	1. 3 2. 2	1. 3 2. 1	1. y 2. y	
As a user I want to be able to see a phylogenetic tree from the genomes.	1. Parse the tree file into JGraphT 2. Display the simple version of the tree graph	1. Boot 2. Boot	1. 2 2. 4	1. 4 2. 5	1. n 2. y	1. JGraphT moved to next iteration
As a user I want to be able to read the graph files.	1. Parse graph files into JGraphT 2. Display a simple version of the graph	1. Vennik 2. Vennik	1. 2 2. 5	1. 3 2. 5	1. y 2. y	
As a programmer I want to set up a specific programming environment with all the desired tools.	1. Set up Travis 2. Set up Octopull 3. Set up Checkstyle 4. Set up Maven 5. Set up Cobertura 6. Set up PMD 7. Define a document flow	1. Oolbekkink & Vennik 2. Hommes 3. Oolbekkink 4. Oolbekkink 5. Oolbekkink 6. Oolbekkink 7. Nieuwdorp	1. 2 2. 3 3. 1 4. 1 5. 1 6. 1 7. 2	1. 2 2. 2 3. 1 4. 1 5. 1 6. 1 7. 3	1. y 2. y 3. y 4. y 5. y 6. y 7. y	

Reflection on Iteration #2

Context project: Programming life
Group: 2

User Story #	Task #	Task Assigned To	Estimated Effort	Actual Effort	Done (y/n)	Notes
As a team we want to deliver our deliverables.	1. deliver sprintplan3.pdf 2. deliver sprintreflection2.pdf 3. deliver productplanning.pdf 4. deliver final productvision.pdf	1. Boot 2. Boot 3. Nieuwdorp 4. Boot	1. 1 2. 1 3. 1 4. 1	1. 1 2. 1 3. 1 4. 1	1. y 2. y 3. y 4. y	
As a team we want to create and maintain our documents.	1. Review and redact the product vision based on received feedback. 2. Review and redact the arch. design based on received feedback. 3. Create a product planning	1. Nieuwdorp 2. Hommes 3. Nieuwdorp	1. 4 2. 3 3. 3	1. 4 2. 3 3. 3	1. y 2. y 3. y	
As a user I want to be able to see the genome graph with edges between the nodes.	1. Draw edges between the nodes.	1. Vennik	1. 3	1. 3	1. y	
As a user I want to have a clear & efficient view of the genome graph	1. Use an algorithm to filter out the crossed lines. 2. Show % of ATCG per node on high-level view 3. Enlarge the scrollbar for bigger graphs	1. Vennik 2. Nieuwdorp 3. Hommes 4. Hommes 5. Hommes	1. 5 2. 4 3. 3 4. 2 5. 3	1. 0 2. 4 3. 3 4. 2 5. 3	1. n 2. y 3. y 4. y 5. y	1. Moved to sprintplan #3

Reflection on Iteration #3

Context project: Programming life
Group: 2

User Story #	Task #	Task Assigned To	Estimated Effort	Actual Effort	Done (y/n)	Notes
As a user I want to see a phylogenetic tree minimap of the active genomes in the graph.	1. Create a view. 2. Display relative tree with lowest common ancestor as root. 3. Highlight the path between all active genomes in the tree.	1. Boot 2. Boot 3. Boot	1. 1 2. 3 3. 2	1. 2 2. 4 3. -	1. y 2. y 3. n	2. Done by Vennink 3. part of the minimap, which we dropped for now
As a team we want to deliver our deliverables.	1. Deliver Sprintplan4.pdf. 2. Deliver Sprintreflection3.pdf. 3. Deliver Productplanning.pdf (final).	1. Nieuwdorp 2. Nieuwdorp 3. Nieuwdorp	1. 1 2. 2 3. 2	1. 1 2. 1 3. 2	1. y 2. y 3. y	
As a user I want to be able to see the difference between 2 paths in the graph.	1. Create selector for 2 paths. 2. Create a difference view.	1. Vennik 2. Vennik	1. 3 2. 3	1. 3 2. 3	1. n 2. n	Replaced by displaying where the bases also occur.
As a user I want to have a clear & efficient view of the genome graph.	1. Show % of ATCG per node on the node. 2. Use an algorithm to filter out the crossed lines (from #2).	1. Nieuwdorp 2. Vennik	1. 4 2. 5	1. 4 2. 2	1. y 2. n	Good algorithm is hard to find.
As a user I want to see what parts of the genome are important.	1. Detect important parts (clarification during meeting) 2. Display important parts in the view.	1. Oolbekkink 2. Oolbekkink	1. 4 2. 4	1. - 2. -	1. n 2. n	Not done due to absence.

Reflection on Iteration #4

Context project: Programming life
Group: 2

User Story #	Task #	Task Assigned To	Estimated Effort	Actual Effort	Done (y/n)	Notes
As a programmer I want our documentation to be up to date so I can see the large lines of the project.	1. Update UML diagrams.	1. Oolbekkink	1. 2	1. 3	1. y	
As a team we want to deliver our deliverables.	1. Deliver Sprintplan5.pdf. 2. Deliver Sprintreflection4.pdf.	1. Nieuwdorp 2. Nieuwdorp	1. 1 2. 1	1. 1 2. 1	1. y 2. y	
As a user I want to see which mutations are present.	1. Display mutations in the graph.	1. Hommes	1. 5	1. 5	1. n	1. No existing solutions found. Has to be done ourselves.
As a user I want to see the genome annotated with information from current sources.	1. Select available sources to use. 2. Transform our current coordinate system into other available coordinate systems so we can find the correct available resources.	1. Nieuwdorp 2. Nieuwdorp	1. 2 2. 3	1. 3 2. 3	1. y 2. n	2. No answer on an email with questions about this topic.
As a user I want to be able to reduce the amount of	1. Select information that needs to be reduced.	1. All	1. 3	1.2	1. y	

Sprint Reflection # 5

Contextproject: Programming life
Group: 2

User Story	Task	Task Assigned To	Estimated Effort per Task (1-5)	Actual	Done	Notes
As a programmer I want our documentation to be up to date so I can see the large lines of the project.	1. Update UML diagrams.	1. Oolbekkink	1. 2	1. 2	1. y	
As a team we want to deliver our deliverables.	1. Deliver Sprintplan6.pdf. 2. Deliver Sprintreflection5.pdf.	1. Nieuwdorp 2. Nieuwdorp	1. 1 2. 1	1. 1 2. 1	1. y 2. y	
As a user I want to have a clear & efficient view of the genome graph.	1. Use an algorithm to filter out the crossed lines (from #2).	1. Oolbekkink	1. 5	1. 0	1. n	
As a user I want to be able to reduce the amount of information (semantic zooming).	1. Create a form in the old control area with a (live) reload button. 2. Filter the information from the graph.	2. Vennik 3. Vennik	1. 4 2. 4	1. 4 2. 4	1. y 2. y	Form is done in graph. Done in branch. Will soon be merged.
As a user I want to load the genome graph using menu items.	1. Make menu items accessible from other controllers. 2. Create a menu item to load the current strains, selected in the tree. 3. Create a shortcut	1. Boot 2. Boot 3. Boot	1. 4 2. 2 3. 1	1. 4 2. 2 3. 1	1. y 2. y 3. y	
As a user I don't want to wait for the same graph to load, when I try to view the same	1. Store last selection of the tree 2. Compare the new selection of the tree to the old one	1. Boot 2. Boot 3. Boot	1. 3 2. 2 3. 1	1. 1 2. 2 3. 2	1. y 2. y 3. y	

Sprint Reflection # 6

Contextproject: Programming life
Group: 2

User Story	Task	Task Assigned To	Estimated Effort per Task (1-5)	Actual	Done	Notes
As a programmer I want our documentation to be up to date so I can see the large lines of the project.	1. Update UML diagrams.	1. Oolbekkink	1. 2	1. 2	1. y	
As a team we want to deliver our deliverables.	1. Deliver Sprintplan7.pdf. 2. Deliver Sprintreflection6.pdf.	1. Nieuwdorp 2. Nieuwdorp	1. 1 2. 1	1. 1 2. 1	1. y 2. y	
As a user I want to know at what position the nodes are that I'm viewing.	1. Think of a way to display it. 2. Implement the visualization of the user's position in the graph	1. All 2. Boot	1. 3 2. 4	1. 2 2. 4	1. y 2. y	2. Base laid out by Boot, rest done by Vennik.
As a user I want to be able to collapse nodes that are of less importance.	1. Make nodes collapsable	1. Vennik	1. 2	1. 5	1. y	Not the only thing I did. Something fell of the sprintplan. Also create first level of semantic zooming, current position indicator, weighted edges and did a lot of refactoring.
As a user I want to see the complete strains on the initial screen	1. Think of a way to fit all nodes on the first screen. (what information to display)	1. All	1. 3	1.	1. y	

Sprint Reflection # 7

Contextproject: Programming life

Group: 2

User Story	Task	Task Assigned To	Estimated Effort per Task (1-5)	Actual	Done	Notes
As a programmer I want our documentation to be up to date so I can see the large lines of the project.	1. Update UML diagrams.	1. Oolbekkink	1. 2	1. 2	1. y	
As a team we want to deliver our deliverables.	1. Deliver Sprintplan8.pdf. 2. Deliver Sprintreflection7.pdf.	1. Nieuwdorp 2. Nieuwdorp	1. 1 2. 1	1. 1 2. 1	1. y 2. y	
As a user I want to highlight paths in the graph of certain strands.	1. Retrieve subgraph from existing Newick tree. 2. Select the strands that you want to highlight 3. Highlight the paths of the selected strands.	1. Boot 2. Boot 3. Nieuwdorp	1. 3 2. 2 3. 3	1. 4 2. 2 3. 3	1. y 2. y 3. y	3. Vennik
As a user I want to see the annotations & known mutations	1. Connect the annotations & known mutations to nodes in the graph 2. Display the loaded annotations in the graph	1. Nieuwdorp 2. Boot	1. 4 2. 4	1. 4 2. 4	1. y 2. y	
As a user I want to quickly locate current view in the overall graph.	1. Draw a position bar at the bottom. 2. Show mutations in the position bar.	1. Vennik 2. Hommes	1. 4 2. 3	1. 4 2. 3	1. y 2. n	2. moved to next iteration.
As a user I want to be able to see the entire graph at once	1. Create more filters. 2. Create filters based on mutations.	1. Vennik 2. Vennik	1. 5 2. 4	1. 5 2. 4	1. y 2. n	2. moved to next iteration.

Sprint Reflection # 8

Contextproject: Programming life

Group: 2

User Story	Task	Task Assigned To	Estimated Effort per Task (1-5)	Actual	Done	Notes
As a user I want to be able to see the entire graph at once and use semantic zooming, based on 'interestingness'.	1. Create a zoom level that show the whole graph at once. 2. Come up with a way to calculate an interestingness factor for the graph. 3. Collapse nodes based on the interestingness factor and a threshold set by the zoom level.	1. Vennik 2. Oolbekkink 3. Vennik	1. 4 2. 2 3. 4	1. 4 2. 2 3. 4	1. n 2. n 3. n	1, 2, 3. will be done by bubble based zooming in the extended time. It's partially done.
As a user I want to see the known mutations that cause resistance	1. Connect the known mutations to the graph. 2. Display the loaded annotations in the graph. 3. Display the annotations in the locator (minimap bar).	1. Boot 2. Boot 3. Boot	1. 2 2. 3 3. 4	1. 2 2. 3 3. 4	1. y 2. y 3. n	3. Almost done
As a team we want to be at the right level for our final report.	1. See which parts are missing for the final report. 2. Review the current final report. 3. Create a tasklist per team member for the final report.	1. Nieuwdorp 2. Nieuwdorp 3. Nieuwdorp	1. 1 1. 2 1. 2	1. 1 1. 2 1. 2	1. y 1. y 1. y	
As a team we want to deliver our deliverables.	1. Deliver Sprintplan9.pdf. 2. Deliver Sprintreflection8.pdf.	1. Nieuwdorp 2. Nieuwdorp	1. 1 2. 1	1. 1 2. 1	1. n 2. y	1. Not required