1. Introduction
2. Overview
3. Software Engineering Reflection
4. Description of Developed Functionalities
5. Interaction Design Section
6. Evaluation of Modules and Failure Analysis
7. Outlook

1 : Introduction

The problem presented in the context was that genetic researchers do not have an intuitive tool that allows them to visualize and compare data extracted through gene sequencing. Such a program would help them identify the segments of a gene sequence that. Specifically, the genetic researchers at Broad Institute and KwaZulu Natal Research Institute requested a tool that is meets the following requirements:

1. Enable us to interactively explore a sequence graph
2. Provide semantic zooming to enable useful visual interpretation at various zoom levels from whole-genome to individual mutations
3. Put the graph in the context of the evolutionary relationship between bacteria
4. Have visual encodings for different classes of mutations and the ability to filter on mutation class (indel / SNP / large variants)
5. Identify mutations and determine the type of variant uniformly across the samples
6. Put mutations in the graph in the context of gene annotations

They also requested several features that they wished to be included into the product, but were not per se necessary. The first was that the product, provide visual representation of meta-data associated with samples, such as drug resistance. Secondly, the product could have indications for convergent evolution of variants. Lastly to integrate with other resources, such as literature databases and mutation databases

2 : Overview

The product, named DNA Lab, is capable of displaying any genetic information given to it in the form of a .gfa file. In addition to a .gfa file as input, it is also capable of loading in a .gff file for lineage colorings, a .nwk file for the newick tree of the phylogeny, and an excel file for any metadata.

The visualization is represented as, in the zoomed out levels, as ribbons. This helps to obstruct unnecessary information from the user and makes any patterns in the data more apparent. These ribbons have two levels, one where bubbles are collapsed, and one where the bubbles are expanded. The expanded level allows a more detailed visualization. Zooming in further through the expanded view will show the standard graph view with nodes and lines between them.

The product also contains a tab within the program with a phylogeny of the genomes that are present within the visualization. The phylogeny and the visualization are connected by the ability to select and deselect genomes in de phylogeny and have only the selected genomes

appear in the visualization. This allows the user to only compare genomes that are pertinent to their interests at that time.

Furthermore the product has a visual representation of the annotations that appear in representative genome from the dataset. This representative genome can be highlighted at the user's discretion, allowing an easy way to see pertinent information about what annotations are present in which genomes and which are not.

3 : Software Engineering Reflection

There are serveral aspects to the software engineering aspect of this project. The first is the use of SCRUM throughout the project and using it to allocate our time and regulate tasks. Second are the tools that we used for continuous integration and static analysis. These tools included github, travis-ci, checkstyle, PMD, and Findbugs. Lastly, the design pattern we chose to adhere to throughout the majority of the project: the model view controller design pattern.
By using SCRUM we were able to efficiently manage our time and tasks during each of the week long sprints. During the first couple of weeks of the project we did have some trouble with our time estimations because we had never done many of the tasks that we had to do so we only had a very rough idea of how difficult they were. But as time went on our estimations became better and better as we became able to apply our past experience to the estimation as well as doing a little bit of research to quickly find if anything that could become an obstacle during the sprint. Throughout the project, tasks such as division of labor was fairly straight forward; If someone wanted to do something, they were usually assigned to it, otherwise people were assigned according to their strengths.

The tools that we used during the project were the tools that were recommended in the guidelines. The use of these tools helped us with keeping our code readable and, for the most part, free of convention errors. There were times when we disagreed with the warnings that some of the tools were presenting us, but every time that happened we were able to rewrite the code in such a way that both us and the tools were happy with the outcome. During the start of the project we had a hard time keeping the code up to date with the warnings that the tools were giving. But just like with the time estimations, as time went on we improved to the point where there were no longer any warnings or errors.

The design pattern that we chose to implement was the model view controller design pattern. Using this design pattern we were able to create a well-designed and maintainable code base. We chose this design pattern mainly because it was the design pattern we found most useful during the Software Engineering course we had earlier. This is because it is designed in a way that makes adding additional functionality to the program follow a basic workflow. It also helps separate out the different aspects of the project, which is helpful for avoiding god classes or something of the sort. There was one week where we were pressured

to implement more than we would normally be capable of and as a result we had to forgo our usual method of working in favor of a more hackile approach. But thanks to the design pattern that we had already been using, even though we had to take a different approach to the work, we were able to complete it and still maintain a workable code structure.

4 : Description of Developed Functionalities

Of all our functionalities, the obvious frontrunner would be the actual visualization itself. This functionality has several aspects to it. The first is the collapsed view where all low level bubbles are collapsed before the visualization is displayed on the screen. The second is the expanded ribbon view. In this view all bubbles are expanded and the data is represented as ribbons. The third is the normal graph view that consists of nodes containing sequence data connected together in a graph structure. Transitioning between these levels is possible by zooming in and out in the visualization, this represents our semantic zooming functionality. At the same time navigating through the visualization is possible by both zooming and dragging the screen to move through the data visually. Each separate level of the visualization has lineage coloring as well. This is calculated by the majority of genomes that pass through either a ribbon or a node.

The vertical spacing of both ribbons and the nodes on the graph represents how 'popular' that path is, which is also represented by how thick the line is. The further away from the norm, in other words how long the ribbon is or how far away the node is vertically is an indicator as to how many genomes pass through that specific ribbon or node. The higher or lower vertically it is, the less popular it is and the thinner the line will be.

Any patterns behind the colorings become more apparent with the use of another one of the programs functionalities: the phylogeny. This functionality shows every genome present in the provided dataset from the given .nwk file. Just like the visualization, lineage colorings are present here as well. By using the phylogeny the user is able to select and deselect the genomes that appear in the visualization. This allows for a certain degree of customization within the visualization.

The program also implements an annotation viewer, which allows the user to see the annotations inside the representative genome in the dataset. <**UNFINISHED SECTION**>

These functionalities were able to come to realization thanks to one of our other functionalities, the embedded database. The base for the embedded database was created by the H2 Database Engine, which we then used and applied to our project. Thanks to this database we are able to vastly increase the amount of data that the program is able to take in because it is all parsed into the database for storage instead of using the memory that is provided to the JVM. This also means that once data is parsed into the database, it will remain

there even if the program is closed; allowing preprocessing to only occur once for every dataset.

The last two functionalities that our program offers is the identification and visualization of both SNP's / indels and different mutations. <**UNFINISHED SECTION**>

5 : Interaction Design Section

Research question

For the interaction design part of the Context Project, we set out research the usability of our application. We want our application to be easy to use and not force users into a complex learning curve before they can properly make use of it. Furthermore, we want it to work fast, so that users do not waste time waiting for application to finish processing data, and we want there to be enough functionality to satisfy the user's needs.

We set up the following research question: "Does the usability of our application satisfy average users' needs?". To answer the research question, we designed a procedure that the users had to follow that would test this.

Procedure

This procedure involved a number of use case scenario's; a sequence of tasks that DNA researchers would likely carry out on a regular basis, would our application be used by them.

These scenarios included:

- Load a new database file
- Have only the first genome of the phylogenetic tree displayed in the graph.
- Have all LIN2 genomes of the phylogenetic tree displayed in the graph
- From the top level view, zoom in on a particular segment
- Highlight all mutations
- Indentify Virus DNA integrated into Tuberculosis DNA

For performing these scenario's, we contacted a group of 5 ex-Biology Master students from Leiden University. Their backgrounds (and initials) are as follows:

- A female Biology teacher (N)
- A male DNA researcher with also a master in Computer Science (B)
- 3 male Biology teachers (R, K, J)

B and the male teachers (R, K, J) are also fervent online gamers with quite some experience with computers. The female teacher (N) has little experience with computers.

Results

The results are as follows:

<**UNFINISHED SECTION**>

1. Table with times for each person to complete each task)
2. Graph mapping the times for each task for each person as lines, with 1 extra line representing the average time

(Data for these 2 parts is still being collected)

(Paragraph about a think-out-loud interview with the users)

Conclusion

(Paragraph about which conclusions we can derive from this data)

(Paragraph about the limitations that this small investigation entails)

(Paragraph about what design changes can recommend for our application, based on the results and the feedback from the users)

6 : Evaluation of Modules and Failure Analysis

First, the visualization module is among our most polished due to its prominence in the program. Both the user interface aspect and code structure of the visualization were where our team spent the most time by far. The ribbons are well done and serve their purpose in simplifying the visualization. The fact that they also have the lineage coloring makes them both more visually appealing and more meaningful to the researchers using the program. The spacing and varying thickness of the ribbons is something that is very nice to have present because it makes it easy to find even more meaning from the visualization.The graph is not anything spectacular because it is the standard. It does have different visual indicator for things such as path popularity and node sequence length, but that is the extent of it.

For the graph, we would have liked to make each node interactive and give more information about itself, but failed with this wish due to time constraints created by other more important implementations. This problem is the main cause for our failures to implement several of the functionalities that we had originally wanted to have.

The phylogeny is, much like the graphical level of the visualization, fairly standard. Something that is different than some other groups is that it only shows the genomes that are present in the visualized data. This is nice because it hides superfluous information from the user and simplifies the program. The phylogeny has interactivity with the visualization in that you can use the tree to select and deselect genomes to be visualized. We chose this instead of a

list or checkbox style because it simplifies the process for the user and is more visually appealing than either of those alternatives. There is a failure with the phylogeny with how we had wanted to visualize metadata <**UNFINISHED SECTION**>

The annotation view is capable of showing the user where the annotations from the representative genome appear in contrast to the other genomes that are being visualized. <**UNFINISHED SECTION**>

The embedded database is well implemented and works almost better than expected. We chose to go with an embedded database as opposed to say, saving data into a text document or using xml parsing to save data, is because a database can be queried. The querying functionality is very useful for us because we have to do a fair amount of specific lookups during both the preprocessing and visualization sections of the project. The reasoning behind chosing an embedded database is because we wanted to make the requirements to run the program as minimal as possible. By using an embedded database, the program can be run on any capable computer, anytime, anywhere, without setup being needed.

7 : Outlook

For future implementations, developers (or ourselves) would be able to use the model view controller design pattern to implement any additional major functionality. The code itself is well documented, so even if there is no formal explanation of it, even outside developers should easily be able to understand what does what and why.

As for functionalities that we would like to implement, we would like to see more interactivity with the graph level of the visualization and the user; perhaps the ability to search for specific nodes in the graph and then be able to select them and view more detailed information about that specific node. The implementation of this would not be terribly hard in theory. Each node would need a unique id that the search function can use to locate it, and then each node would have to include a listener that would either listen for mouse hovering or clicking and then display the information accordingly.