DN/App

Context project - Programming life

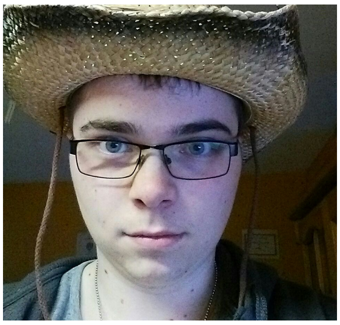
Final Report

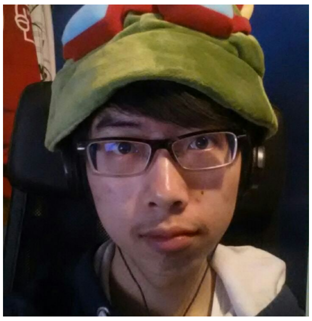
TUDelft

Justin

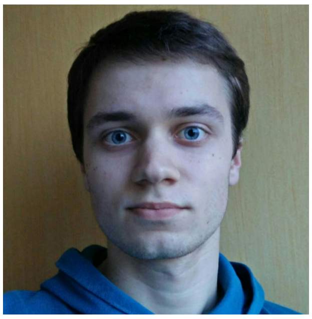
Marissa

Mark





Chak Shun
Scrum Master



Maarten
Product Owner

Team PL1

Chak Shun Yu	csyu	4302567
Justin van der Hout	jrtevanderhout	4319982
Mark Pasterkamp	mpasterkamp	4281500
Marissa van der Wel	mmvanderwel	4323602
Maarten Flikkema	mhflikkema	4306538

Table of contents

1. Introduction

The newest developments in the field of DNA sequencing have made the process of DNA sequencing faster and cheaper. This means a lot of raw data is generated by biologists, which needs to be analysed. To help analyse this data, biologists are interested in ways to detect variations in sequences between organism. To detect these variations, a 'multiple sequence alignment' can be constructed. These multiple sequence alignments can be represented by graph structures, but there is currently no software available to visualise these large graph structures. This is why we created DNApp.

DNApp will however not only enable you to load in large graph structures. It will represent the genome architecture in a way that is most interesting to the user. To create this representation, external information such as mutations which cause drug resistance will affect the graph representation.

DNApp will also allow you to semantically zoom through the graph, collapsing less important nodes to prevent clutter. This semantic zooming is highly interactive, and is based among others, on gene annotation and drug resistance. The customer can also influence what information they value the most, more about this in section 4. It will also contain a minimap so that it is easy to keep track of where you are in the graph. Lastly phylogeny is used to influence the graph structure and allows for highlighting of specific subsets of genomes.

2. Overview

DNApp is capable of loading and visualising specifically generated graph data, phylogenetic trees, gene annotation and known resistance causing mutations.

The nodes within the graph can be selected to show more information about its contents. This information includes the nucleotides contained in the node and the genomes that go through the node. It is also possible to highlight all the nodes of a certain genome.

We have implemented semantic zoom levels in the graph, which means that depending on the zoom level a different representation of the graph is shown. These different representations will have certain nodes collapsed into each other to show less data and keep the graph overview clear. It is possible for the user to configure the settings and choose the kinds of representation that will be used.

Also, a minimap is included that provides an overview of the graph and in which the drug resistant mutations can be identified. Additionally we give an indication on what nodes are in a gene and we offer the possibility to quickly navigate to a certain gene.

3. Reflection on the product and process

The product

Looking back, the development of DNApp was not always a smooth process.

We made use of the GraphStream library for the visualisation of the graph. While GraphStream first seemed to provide what we needed, it turned out not to be easily extendable. Unfortunately we discovered this beyond the point where we could drop GraphStream and use an alternative, so we had to continue using it.

To speed up the general use of DNApp, we have tried integrating a database using the Neo4J framework. However, after having already spent a lot of time on this, we found it would take too much time to finish it and we decided to drop it.

It might have been beneficial to spend some more time on constructing the general structure of the program instead of focussing on functionality right away. This might have enabled us to recognise these things sooner.

The process

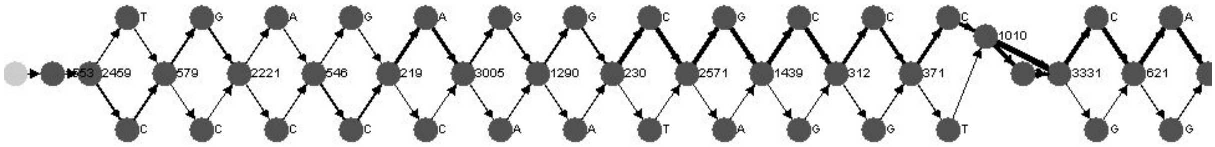
For the overall progress of the project, we made use of Continuous Integration and pull requests on Github. Continuous Integration was used to see whether a commit would considered acceptable. Pull requests on Github had multiple purposes. The first one was to ensure that there would always be a working version of the program on the master branch in which everythings is properly done. The second one was to encourage peer review between team members.

The overall interaction between the team members was good. Sometimes we had some internal problems as a team, but in the end we managed to solve them all ourselves. Every week we also tried to distribute work evenly among all the team members.

4. Description of the developed functionalities

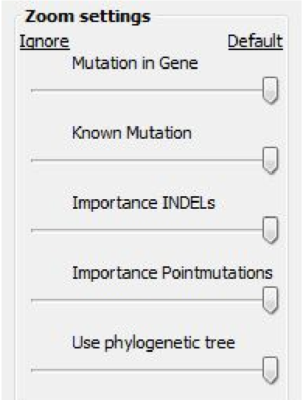
In this section we will guide you through the features DNApp offers.

Firstly, we have implemented the reader and visualiser for the graph. This reader lets you load the provided graph input file. The reader returns a raw graph structure of which the nodes do not yet have a placement. These nodes will then be placed by the node placer, based on the depth level of the node and the highest amount of nodes in any depth level. The depth of a node is based on the maximum amount of edges you need to cross to get to that node from the source. The following figure shows the structure of a graph as it is stored in its input files, without any modifications.



Semantic zooming

Secondly we have provided semantic zooming. Semantic zooming enables the user to hide less important data in favor of visibility. The filtering happens on different kinds of mutations and the user can specify which mutations are more important using the sliders in the option panel (figure on the right). These sliders add a multiplier to that mutation score. When you go to a different semantic zoom level, the score threshold is then changed and nodes with a score under that threshold are then collapsed.


Zoom settings

Ignore Default

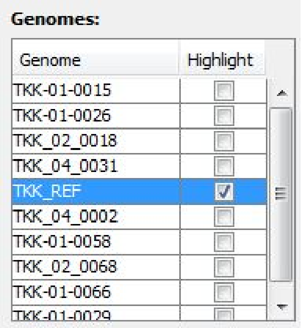
Mutation in Gene

Known Mutation

Importance INDELs

Importance Pointmutations

Use phylogenetic tree


Genomes:

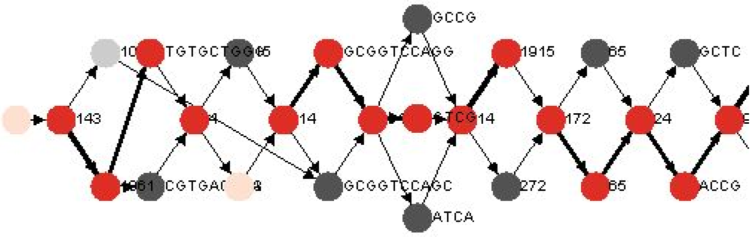
Genome	Highlight
TKK-01-0015	☐
TKK-01-0026	☐
TKK_02_0018	☐
TKK_04_0031	☐
TKK_REF	☑
TKK_04_0002	☐
TKK-01-0058	☐
TKK_02_0068	☐
TKK-01-0066	☐
TKK-01-0029	☐

Genome highlighting

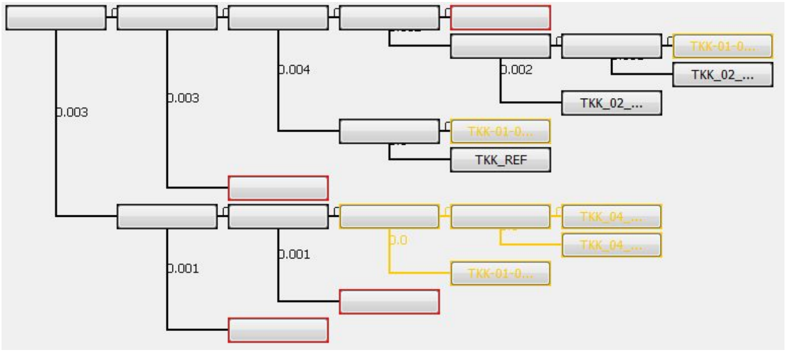
Another important feature is the ability to highlight specific genome strains. When the user ticks the highlight check box in the genome list above in the option panel, the node which contains the checked genome becomes highlighted in red. Another way to highlight sources is by using the phylogenetic tree. When a node in the phylogenetic tree gets left clicked, you can see this part of the tree becomes highlighted. When you click the highlight selection button, the sources selected in the phylogenetic tree will be highlighted in the graph.

Phylogenetic tree

Outside of highlighting sources in the phylotree, it is also possible to collapse nodes in the phylogenetic tree. This is a useful function when a tree containing many genomes is loaded, to enable a condensed view of the phylotree.



The figure below shows a phylogenetic tree in which five leafs, representing genomes, are selected. Those are highlighted yellow. The nodes highlighted in red are inner nodes which children are collapsed.



Minimap

Lastly, we have provided a minimap, which is shown in between the graph panel and the panel showing the content of nodes. The minimap shows where the known mutations are, relative to the reference genome. When a node containing one or more of such mutations is clicked, the nucleotides at the starting position of each mutation inside the selected node is marked light green. Unfortunately, the type of mutation is not distinguishable in this view yet. The minimap also shows where in the graph view the user currently is, shown by the red box.

5. Interaction design

Concerning the interaction design section of our project, we decided to do user tests. In these user tests, we tried to get as much diversity as possible between the people in these user tests. Diversity in gender, in experience concerning the context of our project, in personalities, in the purpose of the program for the person and more.

End-user envisioned

The end-users we envision are biologists using DNApp to analyse many aligned genome strains data and detect variations in the sequence between organism.

User tests

We were able to get in contact with a Biomedical Sciences student at the Universiteit Utrecht. Since she already has some experience concerning genome browsers and is actively involved in the biological field, she was able to provide feedback regarding the interpretation of the content instead of only feedback regarding the usability. This particular interview was very useful for us because the interviewee was highly representative of the end-user that we envision.

She pointed out that a lot of information in the Graphical User Interface didn't have a clear function. She gave us feedback on how we could improve on this and how certains things could be done in a different way. After the user test, we discussed the most interesting topics that arose during the user test and tackled these problems immediately.

One of the things that was unclear for example was what the different colors in the color bar on the left was standing for. After getting this feedback, a legenda was added and the nucleotides in the content of a node showed the content bar were linked with their respective color.

The other interviews we did were with people who did not have any experience in the biological context of this project. Most of the feedback obtained from these interviews was about the clarity of the information provided by the program and how the interviewee could get access to certain functionalities. Basic functionalities such as loading in the needed input files, selecting a node, highlighting a part of the graph and navigating through the graph were clear and straightforward for all interviewees. One recurring problem that was noticed however, was that every interviewee struggled in the beginning to find the difference in highlighting (left mouse button) and collapsing (right mouse button) parts of the phylogenetic tree. Another recurring problem was that the interviewees did not always remember how to access certain functionalities again or what certain aspects of the visualisation meant. To solve these problems a "Help" menu was written which summarises and explains the different aspects of our program and also lists all the possible shortcuts.

6. Evaluation

Looking at the must-have requirements, all of them were fulfilled. However, some of them are certainly better implemented than others. The semantic zooming as it is and the potential of extending it are the parts of the program we are probably most proud of. It is very easy to integrate more information into the semantic zoom levels to make them even more dynamic. Therefore we think this is the best part of the program. The worst of all the must-have requirements is probably the data loading of the graph files. This is because we load everything into memory, which made it very difficult to load the larger data sets. We did try to implement a database for storing the graph and dynamically loading parts of the graph that were visible. Unfortunately we had to drop this functionality due to time constraints and several technical problems, which led to us only able to work with smaller data sets.

A lot of nice-have requirements concerning the visualisation of the graph were fulfilled by us. Highlighting one or more strains as a path through the graph is implemented very thoroughly and is arguably one of the better features of the program. Together with fading colors based on the percentage of unknown nucleotides in a node, line width based on the number of sources going through an edge and specific visual information for collapsed nodes, these were also functionalities we are proud of.

7. Outlook

In nine weeks we created a genome browser capable of comparing multiple genome strains with each other. Unfortunately, in such a short timeframe, we were not able to implement all of the features the customers were interested in. Besides functionalities, optimisation and scalability can still be improved quite a bit. In this section we will share our view on the future of DNApp.

Functionalities

The most important functionality we implemented was semantic zooming. As explained in section 4, we implemented the semantic zooming based on a scoring system. This system applies scores to mutations and is not very advanced at the moment. This means that in the graphs, there exist only a few number of unique scores throughout the whole spectrum of the 10 steps of zoom levels, even if the program integrates user input from the sliders in the option panel. This scoring system can definitely be improved so that it provides a larger number of distinct scores for node groups. More metadata about the genome might be helpful with that, however, with the current metadata that we have we should be able to improve the score distribution. A major aspect to fixing this is having a better understanding of the biological context regarding metadata and how that influences the interestingness of mutations..

The visualisation of the graph can also be improved upon. Due to the software library used for the visualisation of the graph, we only had access to a small amount of graphical aspects, like node color and edge weight. The way we had to handle these aspects was not optimal. A more dynamic and rich graph visualisation would be possible by implementing it from scratch, using a framework such as JavaFX. Such a change to the application would mean a very big portion of the code base will have to be rewritten.

Optimisations

For future expansion on this product, we also see possibilities to optimise it. We would like to implement a faster way to load in the data, especially when it is not the first time using the same data. During this project we had already started on this idea using a framework called Neo4J. Unfortunately there were unforeseen issues with the use of this database system and time was running short, so we had to drop it and focus on more important functionalities The effort put in the database system could still be utilised in the future, since the actual code is still alive in a branch on the GitHub repository.

Appendices

A. Glossary

Branch - A branch is a separate environment of the master branch. Its purpose is for the contributors of the project to make modifications to the code, which can then be evaluated before they are added to the master branch.

GUI - Graphical User Interface.

Metadata - The data representing the phylogenetic tree, gene annotation and known mutations.

Semantic zooming - A variant of regular zooming in which zooming does not only bring an object on the screen virtually “closer” to the user, but also reveals more or less details depending in lower respectively higher zoom levels. Example: Google Maps (shows country names zoomed out, but street names zoomed in).