Architecture Design

Tycho van Heems - tvheems - 4464567
Dex van Leeuwen - dqvanleeuwen - 4475461
Martijn Straatman - martijnstraatm - 4442504
Jochem Tiessen - jtiessen - 4478339
Michael Tran - michaeltran - 4499638

May 5, 2017

Contents

1 Introduction

This architecture design document describes how the final product is structured and how it should behave. The final product is a multi-genome browser programmed in Java. It should allow for visualization and navigation through genome data. This software should be able to run on all operating systems. The main problem is the amount of information that needs to be visualized, since genomes contain a lot of information.
The product is designed for GenomeViz Inc., a company doing research on genomes. By studying mutations in genomes, they can predict traits of living entities.
First, the design goals of the product are defined, where it will be discussed how the product should behave. Then, the software architecture will be discussed, which discusses how the software is structured.

1.1 Design Goals

We split the design goals in four different aspects. First, the performance of the product will be discussed. Second, we will talk about the quality of our code and how the quality is ensured. Third, we will talk about the reliability of our product, which will cover how it's ensured that the product is reliable. And last, we will discuss the availability of the product, which will cover when the product should be available.

1.1.1 Performance

The performance of the product should be good. Our product has to deal with enormous amounts of data, which can severely harm the performance of a product. The product should use an efficient data structure, so all data can be stored without taking up too much memory while keeping it possible to retrieve the data fast. With such a data structure, loading times will still be present, but they should be minimized as much as possible.

1.1.2 Code Quality

The written code of the software should be of sufficient quality, to keep it clear and modifiable. If the code does not meet certain standards, it will be difficult to expand the software.
To make sure the code meet our quality standards, we enforce the use of Checkstyle, PMD and Findbugs. These are programs that keep track of code layout and warns when the written code does not follow the set standards.
Also, all code should be documented using JavaDocs. With this, it will be clear what every method or variable is used for and how it works exactly.

1.1.3 Reliability

The user of the product should always be able to rely on the product. This means that there should be as few bugs as possible as these will cause distrust in users. Also, bugs can greatly hinder the users work with the product, which is not acceptable.
The amount of bugs are minimized by writing automated tests, which check whether most components of the product work as intended. These automated tests should be written for each component as soon as it's made, so it won't be forgotten.

1.1.4 Availability

There should be an available version of the product every week, so the customer can see and evaluate the progress. Also, it's important for the customer, so the customer can give feedback and tell us what he exactly wants next in the product. This feedback can be enforced for the next version of the product.
At the end of the development, there should be a final release, which should be made available. The customer should be completely content with the final release as the customer had continuous opportunities to give feedback and tell us his wishes.

2 Software Architecture Views

In this section we will explain that the application is divided in different subsystems, what these subsystems are and how they communicate with one another.

2.1 Subsystem Decomposition

A user accesses our application through a client. The client can read files with the .gfa extension and uses an embedded database to store the information from aforementioned files. The application then visualizes this information in the form of a graph.

- Parser
  The application contains a parser that reads information from a given .gfa file and stores it in an embedded H2 database.
- Database
  An embedded H2 database is used to efficiently store the information the parser passes on, by using this database the application can function without requiring unfeasibly large amounts of memory.
- GUI
  The GUI consists of a console and a graph, the graph is dynamically drawn by retrieving the required information from the database, and the console provides the user with extra information regarding the graph they are looking at.

2.2 Hardware/Software Mapping

All functionalities of the application are embedded within the client, as a result the client does not need to be connected to a server and can function without an internet connection.

2.3 Persistent Data Management

The application will have persistent data over different sessions because when the application is closed, the data of the last session will remain, like a cache would, and be displayed on the UI.

2.4 Concurrency

As graphs will be visualized from data, that is saved in a database, multiple users should not be able to access or modify the same data at the same time. Should this happen, a deadlock may happen. To prevent this, only one user will be allowed access to the data if a request has been made by the user.

3 Glossary

- Genome
  A genome is a piece of DNA consisting of a combination of a number of bases. There are 4 different bases in a genome; Adenine, Cytosine, Guanine and Thymine.
- Java
  Java is an object-oriented programming language created by oracle, which is mainly used for creating software.
- H2
  H2 is an open source relational database management system that is written in Java. H2 can be embedded in Java application.

References