Emergent Architecture Design

Programming Life Group 3: Thunder Wolf Squadron

Contents

1 : Introduction

This document provides a general overview of the system that is going to be built for our programming life context project. The systems described are going to be described using high level terms and views, meaning that the exact inner workings of the system will not be described, but instead will be presented as a generalization of what is happening behind the scenes.

2 : Design Goals

2.1 : Availability

The product is made in a way to where the client does not have to download or install anything extra to make the project work. This also means that it will be able to be used on any computer that has java installed on it without installing anything extra. We decided to do this because we know that the customer is going to be a genomic researcher and that they may not be able to install anything extra on their system. The only requirement is that the program is in the same directory as the database file that it uses.

2.2 : Manageability

After the creation of the initial working version, our goal is to refactor the code base so that it is accessible and understandable to developers that were not part of the original project. This does mean however that the client themselves will probably not be able to add or change features to the product without a moderate knowledge of computer science and programming.

2.3 : Performance

The product works with an embedded database so the connection with the database should never be a point of concern. Likewise the speed of fast the program can access the database is the fastest that can be possible. The main performance issue comes from the calculations from the data that are used in displaying the visualization. This is going to be made faster with multi threading and good use of preprocessing by taking advantage of our embedded database design.

2.4 : Reliability

The system is very reliable because it is entirely self sufficient, not having to connect to any external sources for data or processing. The only problem with reliability comes with the possibility that the database file could become corrupted and would cause the program to fail.

2.5 : Scalability

The program scales linearly, in principle. The main point of concern for scalability would be with the number of links between nodes, seeing as that can get out of hand rather quickly. Though that relies entirely on the data that is input into the program, so there is not much that can be done in the way of a safeguard from this.

2.6 : Securability

The program itself does not have any security integrated into it. So in theory anyone who gets their hands on the db file can access the data. This means that the security of the data is left up to the user to decide and set up.

3 : Software Architecture Views

This section goes into more detail about the exact design of the system, though it will remain fairly high level throughout. It will describe the relations between systems inside the program and will explain any dependencies the systems have. Subsystems will be explained in the same manner.

To elaborate briefly on how these subsystems interact with one another. Thread 1 handles the splashscreen, this is the popup that the user sees when they first load the program. This has listeners that listen to the progress from the parsers and database and conveys that information to the user through the use of a loading bar that the SplashController is responsible for.

The Parsers are responsible for taking in the data from the files provided by the user. The information that they take in and then parse is then sent through the parser into the database through the DatabaseProcessor.

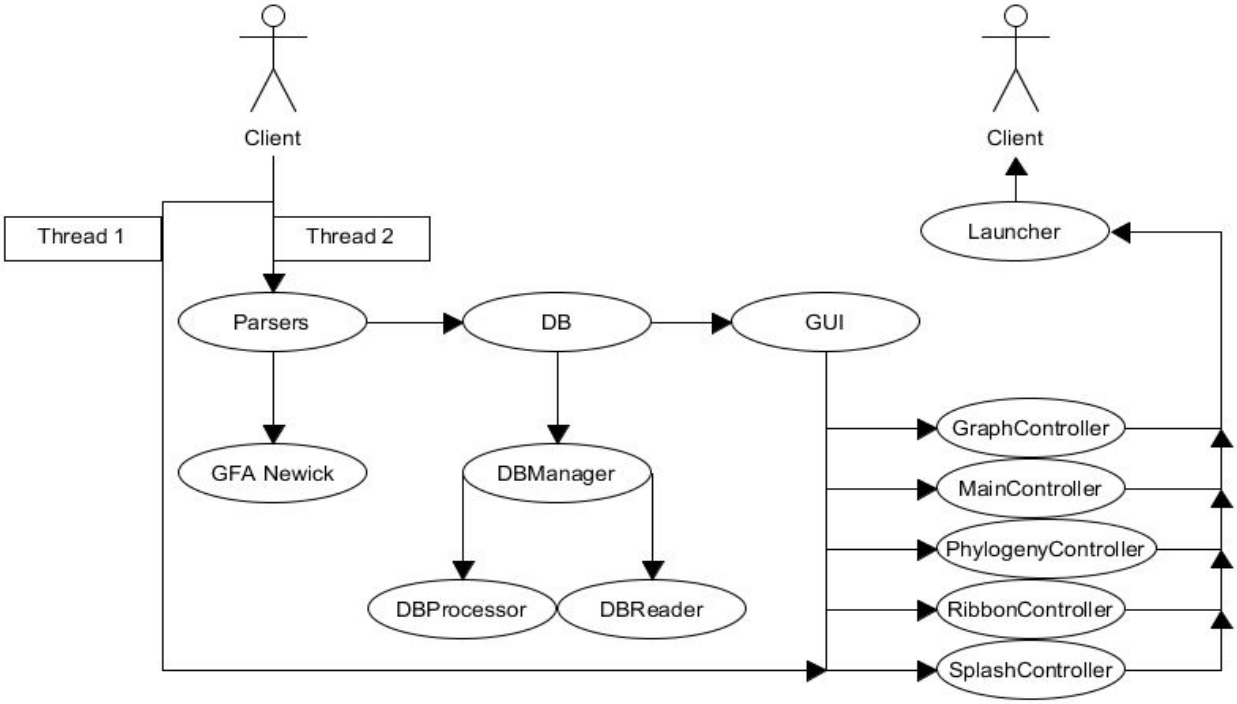
Then the GUI uses the DBReader to read information our of the database through the use of SQL queries. These queries provide the information that is needed to make the gui, created by the various different controllers.

It is important to note that the code base utilizes the model view controller design pattern. In the explanation of subsystems, the controllers are the only thing referenced to

because the controller is in charge of the model and the view, neither of which should be seen as their own subsystems.

3.1 : Subsystem Decomposition

The architecture can be divided into four main systems, the parser, the database, the gui, and the launcher. Subsystems can be seen as the parsering systems, the database manager, and the controllers. The system uses seperate threads to handle different tasks. Thread 1 is responsible for handling the splash screen and loading bars that the user sees when they start up the program. It acts as a sort of reassurance that the program has started



and is indeed working. Thread 2 is responsible for handling the preprocessing and parsing of the data. Thread 2 sends its progress to thread 1 so that thread 1 can update the loading bars and such.

Parser

The parser is the system that takes in the data from the user and parses it to be put into the database. It takes in the information for both the phylogenetic tree and the genome samples and is able to parse both and send them to be stored in the database.The parsers are what make it possible to use the database. Thanks to the database we are able to achieve our design goal of keeping the program accessible to those who are not necessarily well versed in technology.

Database

The embedded database that the program uses for data storage. This system uses the information it gets from the parser and server the purpose of answering any queries that are given to it from the main program. The database, as was mentioned before, is the main factor that helps us keep the setup effort next to nothing and the product accessible to everyone.

GUI

This is the system that is responsible for displaying the visualization to the user. It also calculates the elements for the visualization such as the coordinates of the nodes and lines and where on the screen to draw them. It is also responsible for zooming, but the zooming functionality is not considered a subsystem.

Preprocessing

The preprocessing happens in thread 1. Here is where the data is parsed into the database and all the gui elements have their properties calculated. After this is done, the program uses the data from the preprocessing for the visualization. The preprocessing is responsible for keeping the performance high because it means that the majority of calculations need to only happen one time for every dataset that the user gives to the product.

Controllers

The ribbon and graph drawers are responsible for using the processed data from the preprocessor to draw the visualization on the screen. In the end this is what the user is going to see and use.The controllers, along with their views and models, are what keep the project maintainable. The design pattern means that we are able to keep the project as a whole well organized and functional.

3.2 : Hardware / Software Mapping

The software and hardware don't need to have any special cooperation because we are making our product self sufficient. This means that the database, because it is embedded, is started up and created when the user starts the program. Because of this the only thing that is required from the hardware is that the system the program is being run on has java installed. The other requirement is that the database file is in the same directory as the jar file. Otherwise, the program can not access the database and will therefore be unable to run.

3.3 : Persistent Data Management

The data management is going to be handled entirely by the database. The database, being embedded, will not be running if the program is not also running. This does mean

however that the database cannot be changed without the program running, so there is very little chance for conflicts and also means that each instance of the application (on two different computers or for two different users) have their own instances of their respective data that they use. In addition, the majority of the preprocessing only needs to happen once for any particular database. After that the data is in the db file and the program can simply skip the parsing and some processing and go straight to visualization.

4 : Glossary

GUI - graphical user interface, basically anything that the user sees and interacts with while using the program.