TI2806 Contextproject

Emergent Architecture
Group HI4

April 30 2015

1 Introduction

This document describes how the architecture of the system looks like that is being built during the context project health informatics group 4 in study year 2015. An high level overview of the designed system is given and explained in chapter 2. Chapter 3 contains a glossary. This first chapter is meant to state the design goals of this project.

1a Design Goals

When designing the product we take the following design goals in consideration:

Generic
We try to make the program as generic as possible, so that the program can be used for all kinds of sequential data analysis, not only for the research that must be done for ADMIRE.

Modular
Our goal is to split the program into different modules. The different modules must be as loosely connected as possible so that for example changes within modules would not affect the GUI or vica versa.

Quality of product
We aim for the highest quality of the product. To achieve such a high quality we build a good architecture for our program so that the code is easily maintained. We write automatic test cases, so that if we make an enhancement to our program it automatically checks that we have not broken another part of the code. We aim for a minimum line coverage of 75%. We have coupled a continuous integration server to our VCS system in order to make sure that the code on the master branch always compiles and passes all the tests.

User friendliness
Another goal is to create an easy to use interface for the user, which is partly dependent on the overall quality of the program. We think that the modularity helps to guide the user in a work flow that works well for sequential data analysis.

Performance
The data analysis should be performed within a reasonable period of time with a limit of one minute. When the results are calculated, the output should be shown directly on screen to ensure that the users dont experience long loading times. This means that the program needs to have a responsive design. Also, a form of error management including error prevention and error correction (with error messages) will be implemented in all layers of the architecture.

Scalability and flexibility
The program will be able to process large sets of data with sizes in the gigabyte range. It can handle different types of files and different delimiters.

Use of Design Patterns
We want to implement our application following the Model-View-Controller (MVC) pattern to separate the backend logic from the frontend that represents the program. This enables us to divide domain objects from the GUI elements to keep the code cleaner and the system more maintainable.

2 Software Architecure Views

This chapter describes what our software architecture looks like. First the subsystems are identified and explained. Then the software to hardware mapping is explained, followed by how we store persistent data. Lastly concurrency between information is explained.

2a Subsystem Decomposition

The GUI is split up in 5 different tabs: Import, Link, Specify, Analyse and Result. These tabs represent the different subsystems of the software.

The main goal of this layered structure is to use the data of the inner layer as the input for the next layer. This way it is easy to separate all the modules. All subsystems perform a specific task to evaluate the data one step further.

The innermost layer is the import layer. In this layer the datafiles are specified and read. Using this module the user can specify how the files should be parsed and stored in the program. Then the specified the input is checked whether it is valid and the files are read and stored in a simple data structure.

The data that is read from the files is given to the next layer, the link layer. In this layer different files are linked together to form one sequential data structure. This sequential data structure can then be used to perform the data analysis on. The operations that need to be done on the data is specified in the next layer.

In the specify layer the user specifies what operations need to be done on the data. This is done using a scripting language that is developed during the project. This layer is able to parse the language into instructions for the next layer. These instructions are entered as in a pipeline: The output of one instruction is the input for the next instruction.

The next layer is the most crucial layer. In this layer, the analyse layer, the instructions are actually executed. This is done in pipeline like fashion. Each instruction, which is parsed in the previous layer, operates on the data sequentially. After the last instruction is executed the resulting data is send to the final layer, the result layer.

This last layer is used to visualise and store the resulting data. Using this last layer it is possible to specify how to store files and where to store them. When this layer has finished executing, all transformed data will be available on the location specified by the user.

2b Hardware/software Mapping

This subchapter describes how the software for this project is mapped on hardware. In this project a program for a single computer is designed. The program also runs as one process. However, the process can consist of multiple threads.

2c Persistent data management

For this application it is not required to store information persistently. The output of the program is stored as a text file. The format of the output is so that it can be entered into a statics tool of choosing.

Another idea is to store the state of the processing, for when the processing was not finished. If the state of the program can be stored in XML for example, the user can pick up their work where they left off.

Lastly it would be useful to store the scripts written in the scripting language when a script is needed more than once. This can also be done in textfiles or a custom file format for this language.

2d Concurrency

This subchapter is meant to describe how concurrency issues are solved. Because it is not possible that the program is used by multiple people or processes at the same time, there are no concurrency issues.

3 Glossary

Layered structure - Also called a Multilayered Architecture; a software architecture that uses layers for allocating the different responsibilities of the program. Master branch - The version of our software application that is always ready for deployment.