Emergent Architecture Design

Paul Bakker, pbakker3, 4326091
Robin Borst, robinborst, 4291972
Matthijs de Groot, mawdegroot, 4171683
Julian Hols, jthols, 4247930
Jan van Rosmalen, javanrosmalen, 4318609

April 30, 2015

Contents

1 Introduction

To implement our product we will have different design goals so we can satisfy the needs of the customer. These are described here below. In section 2 we will describe the software architecture views of our product.

1.1 Design Goals

1.1.1 Usability

The most important design goal that we have is usability. It is important that the user can execute the program really efficiently and in a rapid order. That is the reason that we want to make the program executable via the command line. This allows the user to script multiple calls to the program to get the data from multiple patients. Because the customer has indicated that he knows how the command line call works, we think this is the best implementation to execute the program. This enhances the usability for the analyst as he can process a lot of data really quickly.

1.1.2 Modularity

The modularity design goal means that we will define the individual components well, which leads to better maintenance. Also because we work with five people it is beneficial to have good modularity as we can each work on an individual component. And good modularity also helps us greatly in realizing our next design goal.

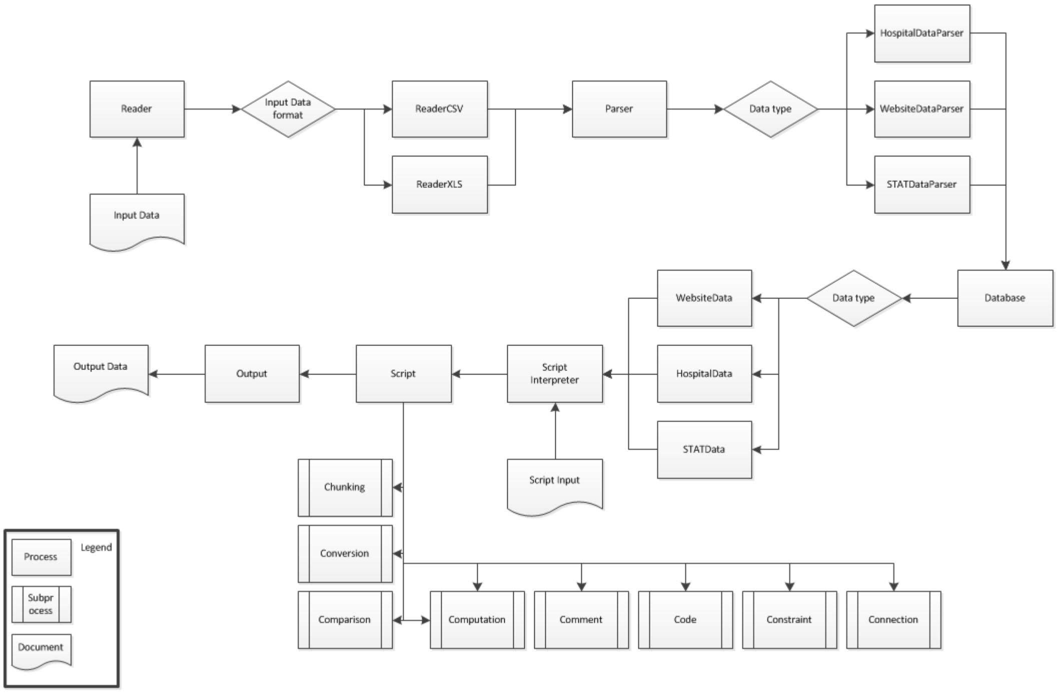
1.1.3 Re-usability

The re-usability design goal means that we will be able to reuse almost all of the code for other customers. We could for example reuse most of the code to easily get the same functionality for other self-monitoring projects.

1.1.4 Compatibility

The compatibility design goal means that the software is able to operate with different file formats outputted by other software products. We will use this to accept multiple file formats for the import and export. The exported files then can be used by different statistics analysis programs.

2 Software Architecture Views



2.1 Subsystem decomposition

We will divide our product into different components, which we will describe below.

- The main class:
  The main is the class that the customer will call using the command line. This will call the correct parser.

- Parser interface:
  The parser will read data line by line from the Reader interface and output this data into a correct data-structure.

- Reader interface :
  The reader will read the data file and outputs it to an inputstream.

- Database class:
  The subclasses of the database class will store all the data during the before and during the data processing.

- ScriptInterpreter class:
  The ScriptInterpreter class will read the script file and execute all the corresponding sequential data operations. It will output a data-structure predefined by us.

- Output class:
  The Output class will generate a text file from the output of the ScriptInterpeter class. The user will be able to select the delimiter.

2.2 Data management

Each data processing operation in the script will take an input data-structure and outputs the same data-structure so it can be used as input for the next processing operation.