Architecture Design

TI2806 Contextproject
Health Informatics

Group A

Technische Universiteit Delft

TU Delft Delft University of Technology

Challenge the future

CONTENTS

1

INTRODUCTION

This document describes the architecture of the application that we are going to build during the context project Health Informatics. Primarily it will provide a high level description of the structure of the software.

1.1. DESIGN GOALS

A number of design goals have been established for this application.

- **Maintainability**
  No matter how robust an application is, it *will* have to be maintained after its initial deployment for a variety of reasons. For example, due to defects that did not emerge during testing or because of changes in the client's environment or even changes in the client's requirements. It is therefor important that the system is designed in such a way that it can be maintained easily.

- **Testability**
  Proper testing allows defects to be found and fixed relatively easily. Therefor the system should be designed in such a way that it allows for effective testing. This can be realized through modular design. The system should not only be modular on a high level, but also on unit level. For example, through the use of interfaces in order to decrease coupling between units.

- **Useability**
  The application will be designed for skilled analysts and so it must be flexible enough and provide sufficient functionality to allow the analysts to transform and manipulate data so that it can be used for a wide variety of statistical analysis.

- **Availability**
  The system will be built in such a way that we will have a demo of the product ready for the client each week. This demo can either be in the shape of an actual working product, or some other way of demonstrating what we have been working on. We work in this way so that the client can experience the changes and additions to the program in as early a stage as possible. So that if the client does not like the new features or would prefer them in a different way, then we can make these changes as soon as possible.

2

SOFTWARE ARCHITECTURE VIEWS

2.1. SUBSYSTEM DECOMPOSITION

The system is divided into five main packages, each with a specific and well defined responsibility. This decoupled design between components allows for easier testing. This section will provide an overview of the different subsystems.

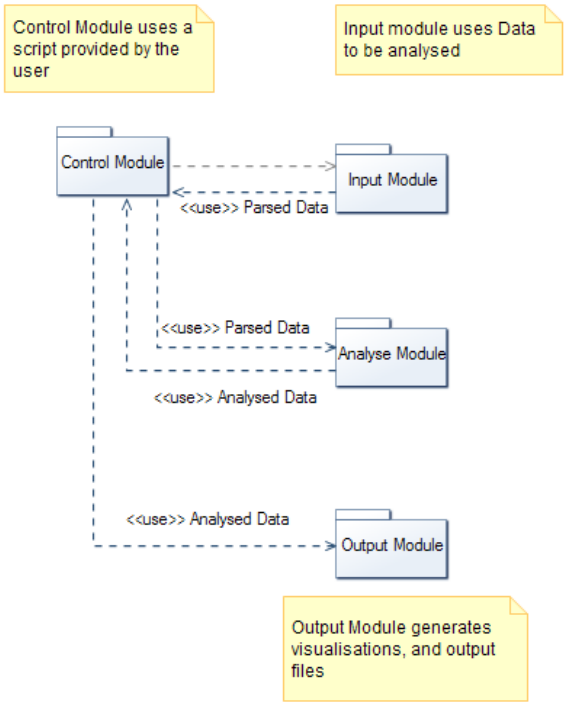

Figure 2.1: The subsystems and the data flow between them

- **GUI**
  The Graphical User Interface allows the user to select the files to be processed, edit the script that is used to transform the data and view the resulting data and visualizations.
- **Control Module**
  The control module is effectively the driving package behind the application. It parses and interprets a script provided by the user to determine which actions should be performed. It then asks the input module to parse the files specified by the user. Then it will ask the analysis module to perform all analysis methods specified in the script on the parsed data. Finally it tells the output module to generate visualizations and formatted files as specified in the script.
- **Input Module**
  The input module will parse input data provided by the user into a software representation that all modules can work with. Input files will require an XML file that describes the layout of the input file, so that if the user would like to use other data, only the XML description file has to be modified.
- **Analysis Module**
  The analysis module provides important transformations that are commonly needed in sequential data analysis.
- **Output Module**
  The output module can generate certain visualizations based on the analyzed data and is

able to output the data to a file in a format as specified by the user.

2.2. HARDWARE/SOFTWARE MAPPING

The application will be a simple desktop application that is ran locally on the user's machine. See figure 2.2.

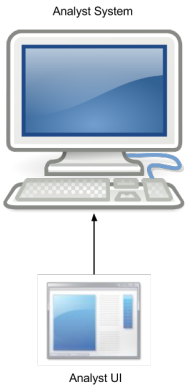

Figure 2.2: Mapping of hardware/software

2.3. PERSISTENT DATA MANAGEMENT

The program will be dealing with three types of persistent data: the input data, output data and visualizations. The input and output data can be either text or binary data. The visualizations will only be binary data. All types of data will be stored and managed by the file system of the user's machine.

2.4. CONCURRENCY

As the application will be executed locally on the user's machine, concurrency between multiple instances of the application is not a concern. However, the software should be designed in such a way that it can be extended to process multiple input files in parallel since it is likely that the user will want to perform the same transformations on an array of input files (e.g. the StatSensor data for each patient).

3

GLOSSARY

- Module = a separate component, which itself performs a defined task and can be linked with other modules to form a larger system. (thefreedictionary.com/module, accessed: 20-May-2015)
- Script = a list of commands that are executed by a certain program or scripting engine. Scripts may be used to automate processes on a local computer. Script files are usually just text documents that contain instructions written in a certain scripting language. (techterms.com/definition/script, accessed: 20-May-2015)
- XML = is used to define documents with a standard format that can be read by any XML-compatible application. XML allows you to create a database of information without having an actual database. (techterms.com/definition/xml, accessed: 20-May-2015)
- Binary data = once a program has been compiled, it contains binary data called "machine code" that can be executed by a computer's CPU. In that case, "binary" is used in contrast to the text-based source code files that were used to build the application. (techterms.com/definition/binary, accessed: 20-May-2015)