Product Planning

Boudewijn van Groos[1], Chris Langhout[2], Jens Langerak[3], Paul van Wijk[4], and Louis Gosschalk[5]

[1]bvangroos , 4229843
[2]clanghout , 4281705
[3]jlangerak , 4317327
[4]pjvanwijk , 4285034
[5]lgosschalk , 4214528
Health Informatics Group C

May 15, 2015

Abstract

This document describes the product planning. It describes the requirements by using the MoSCoW method and by making use of user stories. Furthermore it gives a planning of the releases and it describes definition of done for the backlog items, sprints and releases.

Contents

1 Introduction

When patient has had a kidney transplant, he has to visit the hospital regularly. This is very unpleasant for the patient. Therefore there has been a research, where the patient had to monitor himself instead of going to the hospital (Wang et al., 2013) (van Lint et al., 2014). During the research researchers collected data. Now the data must be analyzed.
We have to create a standalone application that is able to analyze the behavior of the patients. The application must analyze the data for certain events or patterns. Furthermore the results of the analysis must be stored in such way that further statistical analysis is possible. Finally the researchers wants to be able to explore the data by making use of data visualization.
The analysis of the data will be done by making use of Exploratory Sequential Data analysis techniques(Sanderson & Fisher, 1994).

2 Product

2.1 High-level product backlog

In this section we describe the product backlog according to the MoSCoW method (van Vliet, 2007). Therefore the features are divided into four groups. The features are divided based on their priority. Section 3.2 describes for some features a user story. For each feature or group features we will also explain why it has that priority.

2.1.1 Must Haves

These features are essential for the product. Without these features the product is not usable.
The user wants to be able to perform different kind of analysis, so it must be possible to define what analyses is to be perform. If this is not possible the program is useless. Also the user must know how the language work, therefore a manual is essential.

- Language in which the user can describe different analyses
- Executing an analysis that is defined in a file
- Load data in the program based on a description file
- Manual for the analysis description language

The data that is collected during the research, is stored in multiple files. Therefore it must be possible to use multiple files in an analysis.

- Indicating the data connections between the different datafiles
- Load data from different sources using the one data description file

There are different types of data. For example, there are strings, numbers and dates. Not all operations can be done on all type. So it must be possible to distinguish them.

- Indicating the meaning of the various data inputs

It must be possible to do analysis. Therefore the program must implement some data analysis techniques.

- The 8 C's for exploratory data analysis
  - Chunk analysis
  - Comments

  - Codes
  - Connections
  - Comparisons
  - Constraints
  - Conversions
  - Computations

The user wants to use the output of the program in other programs. So this program must be able to output the result in such way that other programs can use it.

- Specifying the output and output format

The user wants that the program can visualize the data in different ways. However not all visualization were equally important to the user. The visualizations are prioritized in the way the user suggested.

- Visualizations from the analyzed data
  - Frequency bars
  - Line graph

2.1.2 Should Haves

These features are very useful. However without these features the product is still usable.

- Visualizations
  - Box plot
  - Stem leaf
  - State transition matrix
  - Lag analysis

When the user wants to use the generated visualizations in a document. He needs some way to create a image of the visualization.

- Exporting the visualizations to images

Some examples in the analysis language will help the user understand the language better. However there is already a manual, so this is a should have and not a must have.

- Implement some example analyses in our analysis description language

2.1.3 Could Haves

These features will only be done when there is enough time.

- Visualizations
  - Histogram
  - Markov chain
  - Transition diagram

When the user want to edit a analysis file, it will be very inconvenient if the user has to edit de file outside the program and then reload it in the program. It would be nice if the user is able to edit the file in the program. Furthermore the user wants to analyze multiple files. It is convenient if it is possible to perform the analysis over all files at once. However both mentioned features will need a lot of time to be implemented. Therefore we consider this as a could have.

- Editor for inputting an analysis description
- Mass input for batch processing
- Preview of the output from the analyses

2.1.4 Would Haves

These features will not be implemented during this project. If this project is followed up by another project, these features might be interesting.
A scripting language for the analysis might be difficult to understand for a user. Therefor the user would profit from a simple GUI to specify the analysis. However this will cost a lot of time to implement and is therefore out of the scope of this project.

- Easy to use GUI for specifying the analyses

2.2 Roadmap

This section will describe the planning for the product. In this roadmap we will plan the major releases. For a detailed overview of the tasks for each week see section 3.2. The numbers of the week correspond to the week of quarter 4. A week ends on Friday.
Week 5 and 7 does not have a specific release goal, in those weeks we should work on the goals for the next week. The goals of week 6 and 8 are too large to achieve in one week. Therefore week 5 and 7 should be used to achieve those goals.

Week	goals
1	Setup project
2	Minimal design of the user interface and basic architecture of the product
3	Product vision and the program must be able to read and write data
4	Product planning and it must be possible to perform the most important data analyses.
6	The user can perform all the types of data analyses and the most important visualizations can be shown
8	The program is able to show all planned visualizations
9	Final product
10	Final report and presentation

3 Product backlog

3.1 User stories

This section describes some user stories. The user stories are based on the requirements as described in section 2.1. Each user story answers the questions who, what and why. The stories follow the following template "As a *who*, I want *what*, so that *why*". The user stories are sorted on priority. The stories at the start have a high priority and the stories at the end have a low priority.

As a user, I want to be able to specify the analysis I want to perform, so that I get relevant results from the program.

As a user, I want to be able to load the data that should be analyzed in the program, so that the analysis is performed over relevant data.

As a user, I want to be able to specify which columns contain dates and in what format the dates are, so that I can perform analysis based on the date.

As a user, I want to be able to specify which columns are numeric, so that I can perform analyses based on numeric values

As a user, I want to be able to specify the name of the data columns, so that I can use those names in the analysis.

As a user, I want to be able to create a file that specifies which files should be read for the data, so that I can load multiple files at once.

As a user, I want to be able to specify the relation between different data files, so that I can use multiple files in an analyses.

As a user, I want to be able write the output of an analysis to a file, so that I can use the result for further analysis.

As a user I want to be able to view the data as Frequency bars, so that I can see how often an event happens.

As a user, I want to be able to view the data as a Line Graph, so that I can see how the behavior of the analyzed person changes over time.

As a user, I want to be able to perform one analysis over multiple datasets, so that I can perform the same analysis over all statsensors at once.

3.2 Initial release plan

This section will describe the planning for the product. The release plan is based on sprints of one week and on the roadmap described in section 2.2. The numbers of the week correspond to the week of quarter 4. A new iteration starts on every Friday. For each week we will list which features the product should have and which additional task must be done. Furthermore for each feature we will list the priority. This is the same priority as in section 2.1.

3.2.1 Week 4.1

- Setup the software that is used during the project
- Obtain the requirements

3.2.2 Week 4.2

- A basic architecture for the product
- A design for the user interface
- A draft version of the product vision

3.2.3 Week 4.3

- A minimal user interface according the design of week 4.2
- The final version of the product vision

- A draft version of the product planning
- The user must be able to specify in a data description file how a file should be read by the program (Must Have)
- The user must be able to specify which data must be written to a file (Must Have)
- The user must be able to perform constraint analyses (Must Have)

3.2.4 Week 4.4

- The final version of the product planning
- The user must be able to perform chunking analyses (Must Have)
- The user must be able to perform connections analyses (Must Have)
- The user must be able to perform computation analyses (Must Have)

3.2.5 Week 4.5

- The user must be able to perform codes analyses (Must Have)
- The user must be able to perform comparisons analyses (Must Have)
- It must be possible to show the data as frequency bars (Must Have)
- It must be possible to show the data as a line graph (Must Have)

3.2.6 Week 4.6

- The user must be able to perform comments analyses (Must Have)
- The user must be able to perform conversions analyses (Must Have)
- It must be possible to show the data as a box plot (Should Have)
- It must be possible to show the data as a Stem-and-Leaf plot (Should Have)
- Input for SIG

3.2.7 Week 4.7

- It must be possible to show the data as a state transition matrix (Should Have)
- Show the data with Lag analysis (Should Have)
- The user must be able to export the visualizations as an image (Should Have)
- It must be possible to show the data as a Histogram (Could Have)

3.2.8 Week 4.8

- It must be possible to show the data as a Markov chain (Could Have)
- Specify multiple files that all will be analyzed individually (Could Have)
- Implement certain analyses functions in our language

3.2.9 Week 4.9

This is the last week where it is possible to work on the code. No new features are planned for this week. In this way we will be able to handle some delay during the process. Furthermore this week is used to repair the last bugs. Therefore there is a feature freeze on Wednesday June 17.

- Final input for SIG
- Draft version of the final report

3.2.10 Week 4.10

- Final report
- Product presentation

4 Definition of Done

In this section we will discuss when a task is considered as done. In general a task is done when there is nothing left to do for that task. We will discuss the definition of done for backlog items, sprints and releases.

4.1 Backlog Item

A backlog item is done if it is implemented as described and it follows the description of the user stories. Furthermore the code should have been tested with unit tests. All the other features should still work and all the tests should pass. The code must be reviewed by at least two persons who have not worked on that specific item. The code should be clear and when needed, it should contain comments. Furthermore the code should follow the languages conventions and it should have clear names for the variables. When the item meets all these requirements, than it is considered done.

4.2 Sprint

Each sprint should have a sprint plan and a sprint reflection. Any deliverable that has a due in or at the end of the sprint should have been made and handed in. Furthermore if needed, relevant documents, such as the architecture design, should have been updated. Critical bugs that are discovered during the sprint must be fixed. If it is not possible to fix them during the sprint, than they have to be solved in the next sprint. Finally all the task of the sprint should be completed as described in the previous section. If it is not possible to complete a certain task in a sprint, than the sprint reflection should explain why it is was not possible to finish the task.

4.3 Release

Each sprint ends with a new version of the product. Sections 2.2 and 3.5 provide an overview of the planned features for each release. Based on that, each sprint will add some new features to the product. A release is only allowed to contain features that are considered done, see section 4.1. Therefore all the features in a release are tested and the code should be proper. Additionally we must test whether the features work correctly together. Furthermore a release may not have any critical bug. Finally for each release a demo has to be prepared and demonstrated.

References

Sanderson, P. M., & Fisher, C. (1994). Exploratory sequential data analysis: Foundations. *Human–Computer Interaction*, *9*(3-4), 251–317.

van Lint, C., van der Boog, P., Schenk, P., van Dijk, S., Romijn, F., & Cobbaert, C. (2014). Zelfmonitoring van nierfunctie na niertransplantatie: de patiënt als regisseur. *Voorjaarscongres Nederlandse Vereniging voor Klinische Chemie en Laboratoriumgeneeskunde (NVKC) 2014*.

van Vliet, H. (2007). *Software engineering: Principles and practice*. Chichester, UK: John Wiley Sons.

Wang, W., Brinkman, W.-P., Rövekamp, T. J., van der Boog, P., Alpay, L., & Neerincx, M. A. (2013). Feedback to renal transplant patients in a self-management support system. *European Conference on Cognitive Ergonomics*.