Product Planning

Health Informatics
Group 4
May 7 2015

Table of contents

1. Introduction

For creating a good product, careful planning is required. This documents provides the prioritized agile product backlog and describes the top items to the team. This helps the team to determine which items they can and should complete during the coming sprint. By moving items from the product backlog to the sprint backlog, each product backlog is divided into one or more sprint backlog tasks so that the group can effectively divide the work and complete their goals. The goal of our project is to create a standalone software application which can be used by highly-skilled researchers to analyse the behaviour of renal transplant patients.

In the second chapter this document first provides an analysis of the program by describing the context, problem and stakeholder inputs. Based on this the steps needed to complete the project will be defined in a roadmap. Here you will find the major release schedule and goals.

Chapter 3 contains a product backlog, which is done by means of user stories of features, technical improvements and know-how acquisition. In chapter 4 a definition of ‘done’ is given on backlog items, sprints and releases. This will enable the group to know when a feature is done and when the corresponding sprint item can be closed. The 5th and last chapter gives a glossary to give you an explanation for any jargon or complicated terms.

2. Product

This chapter gives an high level overview of the product. Paragraph 2.1 gives a brief overview and analysis of the program. Then in paragraph 2.2 this overview is transformed into a roadmap of major release schedule. Here the major releases of the program are given in a clear overview.

2.1 Program overview

For a good planning it is important to know what the end product should do. This paragraph gives an high level overview of the product backlog.

Analysis context

Chronic kidney disease is seen as a major public health problem that is still growing worldwide. At this moment renal transplantation is seen as the best treatment available for patients. The ADMIRE Project - organized by LUMC, TU Delft and TNO - introduces a disease management system for patients self-monitoring. This includes a new home-based medical device for measuring the creatinine and blood pressure levels and a website for feedback on the patient's health status. The data set consists of measurements and website data of a group of 50 patients in the Netherlands.

Analysis problem

Researchers who are interested in the health status and the behaviour of patients performing self-monitoring need the dataset to be pre-processed for further statistical analysis. The data needs to be analysed sequentially per patient in order to discover any events or use patterns such as the type of errors that are made during the entry of measurements on the website, how patients deal with their measurement scheme and with feedback of the system and whether they enter the right values on the website (or false dummy values).

Analysis stakeholder inputs

The application needs to be able to handle different forms of data from three data sources: a portable measurement device, the values entered on the website and hospital administration. The program should be able to import raw data, transform it into other kinds of data and be able to export these sorts of data for further use in other (statistical) software applications. The user also wants to specify transformations that should be performed on the raw data in a scripting language specifically designed for this purpose. This could be done with a script editor or graphical user interface. The tool is required to support the researcher with the exploration of data by showing suitable forms of data visualization.

2.2 Roadmap

This paragraph gives a roadmap for the most important releases of the product. The major releases are spread across different sprints and described below with the corresponding goals. All the releases are continually tested with system testing during the entire project.

Sprint 1: 1st release 01/05/2015

This release will contain at least the following features:

- A filereader that is able to read in files and able to handle different delimiters.
- A generic writer that is able to write the expected output with selected delimiter.
- First version of the GUI containing a menu bar and tabs for the four steps of data analysis: import files, link files, analyse files and print results.

Sprint 2: 2nd release 08/05/2015

This release will contain at least the following features:

- A linking module that groups different files together based on their primary keys.
- Basic implementation of three data operations (chunking, constraining and commenting), with a scripting language construct.
- Product vision document to envision the target customer, requirements, timeframe and budget of the end product.

Sprint 3: 3rd release 15/05/2015

This release will contain at least the following features:

- Extended implementation of chunking, constraining and commenting.
- Further improvement of GUI for writing scripts (in a script editor) based on feedback of the stakeholder.
- Process the linked data into a sequential data structure to be able to group records on their date.

Sprint 4: 4th release 22/05/2015

This release will contain at least the following features:

- Implementation of two more data operations (coding and connecting), with scripting language constructs.
- Implementation of the first form of data visualization: timelines.
- GUI contains a basic version of the graphical representation of our scripting language.

Sprint 5: 5th release 29/05/2015

This release will contain at least the following features:

- Implementation of two more data operations (comparing and computing), with scripting language constructs.
- Implementation of the second form of data visualization: frequency bars.
- Perform user acceptance testing by interviewing the client about the product.

Sprint 6: 6th release 05/06/2015

This release will contain at least the following features:

- Implementation of the final data operation (converting), with scripting language construct.
- Implementation of two more forms of data visualization: boxplots and stem and leaf plots.
- Improvements based on feedback from the user acceptance tests.

Sprint 7: 7th release 12/06/2015

This release will contain at least the following features:

- GUI containing the final version of the graphical representation of our scripting language.
- Implementation of the last two forms of data visualization: two-dimensional time series and histograms.
- Emergent Architecture document presenting the final state of the architecture design.

Sprint 8: 8th release 19/06/2015

This release will contain at least the following features:

- Solved bugs and other problems from the previous sprint(s).
- Final product containing the ability to read files, link them, analyse them using the scripting language and save the output file in the required format.
- Final report about the developed, implemented, and validated software product.

3. Product backlog

This chapter describes the product backlog of our program. The first paragraph gives the user stories of the features that need to be implemented. The second paragraph focuses on technical issues found in existing products of competitors and how they could be improved in our product. The third paragraph describes the user stories for the know-how acquisition. The fourth and last paragraph shows an initial release plan with milestones.

3.1 User stories of features

In this paragraph the user stories of features are sorted by priority using the MoSCoW method:

Must have

These features are must haves since the program would not be functioning and meet the (minimal) requirements of the customer without these implemented.

- As a researcher, I want to read in any kind of data file I select from a directory so that the selected files can be linked and analyzed.
- As a researcher, I want to export textual data and graphs after analysis of a specific patient so that I can save the results in a directory.
- As a researcher, I want to export the results in a form that can be entered easily in another program so that I can use it for further statistical analysis.
- As a researcher, I want to link datafiles on certain primary keys so that they can be used for a given ESDA on one patient.
- As a researcher, I want to perform transformations on the raw data so that I can do a sequential data analysis.
- As a researcher, I want to group records on particular attributes (such as time periods or certain values) so that I can analyze a particular group.
- As a researcher, I want to add labels to created data records, data elements or data chunks so that I can mark them with keywords for sequential data analysis.
- As a researcher, I want to add relations between data records so that I can link data elements which are related to each other linearly, event-driven, qualitatively or through time.
- As a researcher, I want to select data elements based on a condition so that I can perform further transformation on a part of the data and exclude the remaining data temporarily.
- As a researcher, I want to compare time between events (such as measuring time and time of input it into website) so that I can do a sequential data analysis.
- As a researcher, I want to use a simple, intuitive scripting language so that I can specify the transformations that should be performed on the raw data.
- As a researcher, I want to enter the script in a script editor so that I can enter the script in the program.
- As a researcher, I want a modern and easy-to-use interface to select files, link files, enter a script for transforming the imported data and select a directory to store the output.

- As a researcher, I want to visualize the analysed data with timelines and frequency bars so that I can explore the time lapse and counts of certain data elements.

Should have

These features are should haves since the basic components of the program would work without these features but it would be better (meet the requirements of the customer better) if these would be implemented.

- As a researcher, I want to convert data records including type conversions, code combinations or resolution changes so that I can allow new patterns to emerge in the sequential data analysis.
- As a researcher, I want to do computations such as counting, summation or statistical operations on the data so that I can perform exploratory analysis on the data.
- As a researcher, I want to add comments to certain parts of the data such as data elements, data chunks or conversions so that I can make memos of my own reflections on the data.
- As a researcher, I want to visualize the analysed data with boxplots and stem and leaf plots so that I can explore the data.
- As a researcher, I prefer a graphical representation for the scripting language so that I can enter the script in the program.
- As a researcher, I want to import and export the state of the software in XML so that I can describe the structure of the data.

Could have

These features are could haves since they are not necessary to the customer and to the functioning of the program but it would be nice to include them if there is enough time left in our timeframe.

- As a researcher, I want to export the results in different kind of files (other than text files) and with different delimiters so that I can easily adjust the output to the next (statistical) program in which I am going to enter the file.
- As a researcher, I want the program to automatically recognize the structure (columns) and types of data so that I don't have to set this every time when I am importing a new file.
- As a researcher, I want the program to be able to also parse an XML file that describes the input so that I don't need to describe the structure of the data when importing a file.
- As a researcher, I want the GUI to show me (real time) how much progress is made in the analysis of the data so that I can estimate how long I have to wait for the results.
- As a researcher, I want to call the program by a name and recognise it by its logo so that I can share it with others.
- As a researcher, I want a logger to log all the operations I have done in the program so that I can have an overview of the adjustments I made.
- As a researcher, I want to visualize the analysed data with two-dimensional time series and histograms so that I can explore the data.

Won't have

These features are won't haves since the customer doesn't actually need these features to be implemented. There probably won't be enough time left for the following features but it could be interesting for a follow-up phase of the project.

- As a researcher, I don't want the program to have analysis tools since I already have those.
- As a researcher, I don't want to be able to adjust the collected data from the ADMIRE project since the values should stay unchanged for accuracy.

3.2. User stories of technical improvements

Must have

This feature is a must have since it's a main problem in similar existing products that needs to be improved to differentiate our product from other products and to enable the user to do sequential data analysis, specifically for the ADMIRE project. This technical improvement would be a unique selling point for our program.

- As a researcher, I want the program to be flexible with input data and to be able to handle special cases such as symbolic data (date objects), linear data (temporal relations) and encoded data so that I can analyse all the data elements I want.

Should have

These features are should haves since their improvement would add (new) functionality to the program but they are not necessary for the customer.

- As a researcher, I want the program to execute the data transformations in a more automated manner (than Excel, OpenOffice, etcetera) so that I can do more with less (scripting) work.
- As a researcher, I want the program to be easier to use than most data processing frameworks and database systems so that I don't need to spend much time on configurations and learning how to use the program.
- As a researcher, I want the program to be able to compare the actual measurements with the values entered on the website (specialized on ADMIRE data) so that I can deal with data accuracy problems that occur in web help systems.

3.3 User stories of know-how acquisition

This paragraph describes which documents the team needs to read in order to build the background knowledge that is required to make a future technical decision or precise scheduling based on the user's needs.

The project team needs to read about the following subjects:

- As a researcher, I want the development team to read about the so called eight C's discussed by Sanderson and Fisher so that the team gets introduced to different general transformations that I would like to perform on sequential data.
- As a researcher, I want the development team to read about statistical programs such as SPSS so that they will know how these programs work in general and so

that they understand in which form the results of the data analysis should be delivered for further analysis.

- As a researcher, I want the development team to read about the ADMIRE project and research that is already done on the way patients deal with web help systems and feedback of self-management support systems (e.g. the paper of Wenxin Wang et al. on "*Feedback to Renal Transplant Patients in a Self-management Support System*").

3.5. Initial release plan (milestones, MRFs per release)

4. Definition of Done

This chapter describes the definition of done so that every group member will have the same end goals. This will enable the group to know when a feature is done and when the corresponding sprint item can be closed.

Our definition of done focuses on three levels: backlog items (features), sprints and releases.

4.1 Level 1: Backlog items

We consider a backlog item as done when it has an test coverage of at least 75% for non-GUI elements. These tests can be divided into unit tests and other automated tests. For a feature to be merged (using a pull request) into the release version of the product, it has to be approved by at least two other team members. Their approval will be based on the test coverage, code readability, javadoc and the overall code quality. Javadoc should be added on every class and method.

4.2 Level 2: Sprints

Every sprint ends at friday 23.55. At that moment all the items on the sprint plan of the current sprint have to be finished, according to the definition of done for backlog items. A new release of our product should be submitted to the version control server (github) master. The sprint can be closed when the release has been tested completely and the code is approved (as with any feature). Our continuous integration server (Travis) tests if the application runs as intended by running the unit tests. All the unit tests should pass and the system should be improved based on the weekly acceptance test with the customer (Wenxin Wang). There shouldn't be any bugs/errors left in the system and the program should behave and look like the customer wanted. We should also have written a sprint reflection for that sprint and a new sprint plan for the coming sprint.

4.3 Level 3: Releases

A release version of our product is done at the end of a sprint. The release version should contain the features that should have been implemented for the corresponding sprint. This also includes the minimal test coverage and other conditions described in the definition of done for a sprint. It is also important to note that the feedback of the user for that week should be processed before a release is completed.
In the final release we should have implemented all must haves since the program can't function without these mandatory features. Most should haves (50 to 60%) and some could haves (30 to 40%), which are defined in 3.1, have to be implemented. As explained in chapter 3, these features are not necessary for the user and the system to work but our goal is to implement them partially since it would be nice to include them. It would make our program more user friendly.
Lastly, the final release and other major releases should be approved by the stakeholders, the end users and obviously by ourselves too. This means that it should be simple and

efficient to use for the users and well documented and designed by us. The SIG test is also an important part of this. They will determine if the code meets the standards and if it is clearly structured and documented.

As a team we also strive for maintainability and extendability, so that the system can be easily maintained, improved and updated by us or other people after our final release. This will also be taken into account throughout the entire project.

5. Glossary

Travis	A continuous integration server: This is used to test automatically for problems and ensure a working product on the server.
Github	A version control system: This is used to keep track of several versions of the program that are created.
Master branch	The version in our version control system which is always ready for release.
Self-management Support System	A system for self-monitoring and self-management based. It is recognised as one of the critical components in improving health care for people with long-term conditions.