Project experiences

Introduction

This document is a description of the team's requirements engineering work during this project and all the experiences and reflections that followed.

During this project we have used different techniques to elicit requirements and set up our specification. We have also used other methods to prioritise requirements and which features should be in the scope of the product.

We have used reqT to construct QUPER-diagrams, E/R-diagrams, data dictionaries and virtual windows.

Elicitation techniques

Interviewing the client

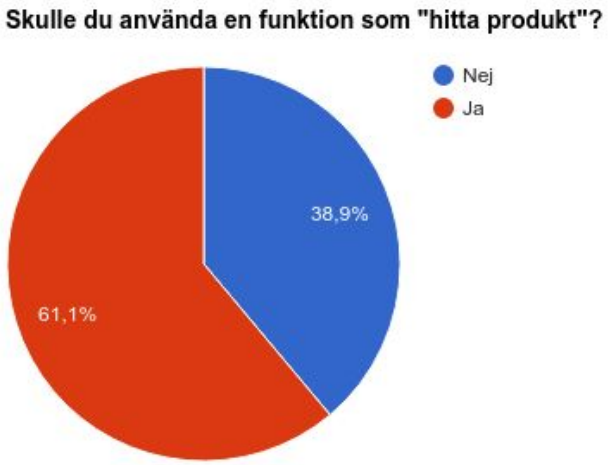
During the whole project timelapse, we have had numerous client meetings to discuss how to make the best product outcome, being as cost- and time efficient as possible.

We started off early to discuss which areas of functions our client thinks is the most important. As time passed, we have tried to be more and more specific within the requirement categories on which requirements that they think are the most desirable. Rather than having a ordinal scale methodology, we tried having a version of a 100 $ approach where we wanted to see which requirements they think are the most and least important, where they also could prioritize some requirements with the same priority.

After each meeting, we discussed internally in our group if the prioritizing our client had made could motivate the time and cost for the requirement to be fulfilled. If we were unsure if the time or cost could be motivated, we prepared data in the form of a survey or other background data to show to our client in order to de-prioritize the requirement.

Survey

In preparation for release two, we sent out a survey to see if the requirements our client had specified were interesting for the target user group.This survey included questions which laid the basis for a lot of the requirement priorities set for the project. We received 19 responses to our survey. Below is a sample of the result created by data from the survey:



This question was if our client's request to have a function to find a specific product was interesting. We chose to include this question in our survey because we knew this requirement would have been too expensive to implement and we got the impression from our client that it wasn't their most prioritized requirement.

After the survey was made, the result were presented to our client in the form of a Powerpoint presentation. One of the points we brought up were that since the survey didn't give enough support to function presented above to motivate the cost, a decision was made to lower the priority on that specific function.

Quality grid

Constructing a quality grid helped us get a sense of which requirements to be prioritized. The different angles of incidence the features of quality grid gave us (Usability, Reliability, Portability) opened new perspectives on the quality requirements.

Study similar companies

We have looked at competitors apps to see if we are lacking some functionality. We think with our solution, it would be best if we would have been in charge of the servers so that they are up and running, which some competitors aren't as good with. For this we could charge a fee.

At the same time as we constructed our QUPER-diagrams we conducted evaluations of competitor products. We decided to evaluate some functionality in our product against whole competitor systems. The evaluated functionality was; navigation, public transportation and parking. The evaluated competitors were search engines, public transportation systems, parking apps and navigation systems(gps).

From the QUPER-diagram we got a better understanding of the market and it helped us construct more and better quality requirements. The result can be viewed in one of the attached html files.

Cost/benefit

We removed some of the original requirements prior to release 1 due to the fact that they were very hard or almost impossible to achieve while their value was highly dubious.

After some analysis we have concluded that the payment option is too expensive compared to what it offers.This is because we would have to create a system that could communicate with each store's payment system which would have been a monumental task and a app on it’s own. This requirement will not be apart of the scope.

The requirement for the app to be able to direct the user to the mall from anywhere is an expensive feature and has a very low value. Due to its low value and the expense and time required to implements what accounts to be an entire global navigation system this requirement has been reduced to a requirement for the app to use the local public transportation timetables to help the user to get to the mall.

It was concluded that the requirement to implement a feature for finding the shortest route through the mall was unfeasible due to the fact that the algorithm would be CPU expensive and be difficult to implement. This could be solved by letting a server do calculations and communicate the answer to the application. But due to the complexity of the problem which actually is a NP-complete problem we have decided to skip this feature.

In addition the shortest route feature also represents a conflict of interests as it is contrary to the purpose of malls to help people get through the mall as quickly as possible as this ruins the chance of impulse shopping. The value of this feature is very low compared to the cost of implementing it and has been discarded.

Removed requirements

A requirement for the app was that it could be used for paying in all the stores without requiring the stores in the mall to use the same payment method. This would resulted in the app having to support very different payment services systems. Uniting them with a single interface would take too long time to implement and would cost accordingly while the value of the payment service is insufficient..

Requirement ID

In release 1 we used automatic requirement numbering (related to the document headers) for requirement identification. If some requirement is changed, needs to be removed, and so on, the identification of the requirement can mess up other dependant documents. Instead

we should perhaps have chosen a different identification scheme, such as a unique describing name for every requirement or by simply giving each requirement a unique number. We have corrected this in release 2 and given the requirements unique describing names.

Customer evaluation of the system

We conducted a survey to see what potential future users would be interested in when using a product like ShopMate. After compiling the results from the survey we evaluated what requirements and features customers deemed unnecessary and found out that some of them were the same features that we decided not to implement after our cost/benefit analysis.

Another thing that the survey showed was that out of the 20 people that participated no one had a smart-watch. The smart-watch requirements was very prioritized by our customer. We brought this to our customers attention and they said “We want the feature but in a later release”. It has changed now and are no longer as prioritized as before.

The result from the survey is attached to the release.

Checklist

Prior to release two we made a checklist. The checklist was created so that our customers can check the requirement specification for errors. The checklist will help our customer validate the requirement specification regarding the document structure, sufficient descriptions and definitions, lack of requirements, vague formulation and the overall content.

The format of the checklist was chosen because of the structure, which we thought was straight forward; it was easy to overlook and to understand. The checklist was based on a standard requirement specification checklist, which we adapted to fit the project we were working on.

Result

Overall, the major changes that have been done since release one is prioritization and further development of the system requirements. By evaluating the cost against the benefits, surveying the target user group and by continuous discussion with our client, we have been able to prioritize the requirements accordingly.

For release two, we have chosen describe our data requirements more specifically with different methods and techniques by using Quper, generating data models, data dictionaries, virtual windows and E/R diagrams. We have used the requirement tool reqT while developing the data requirements further.

While developing our requirements further, we have chosen to categorize our requirements differently than the last release. Our new categorization divides the requirements into different levels, such as goal, domain, product and design.

To ensure continuously work of improvement, we have produced a quality grid and a validation checklist. By letting our customer overlook the work we have done while it is in progress, we will reduce the chance of misunderstandings and friction. We will also be more sure that our client will be more satisfied upon delivery.

Lessons learned

Since the previous release, we have learnt a lot thanks to the feedback we have received from both our mentor and customer. Many of the things we have learnt and implemented have been covered under the results section, so here are a few bullet points of what we have learnt:

- Giving requirements IDs related to their field and category and avoid numbering them.
- Using reqT for data modeling and drawing diagrams,
- Being able to structure requirements better.
- Elicitation and prioritizing what the client think is important.
- Learning more about release planning.
- Don’t do a release plan manually, we did when reqT crashed for the hundredth time, it is both time consuming and probably cause mistakes that can cause the entire operation to be repeated.

Team member contributions

Daniel Dornlöv

Validation checklist, Release Plan, renaming of requirements

David Cartbo

Validation checklist, Release Plan, renaming of requirements

Jonathan Lundholm

Elicitation of additional requirements, structuring the requirement specification document and transferring it into a LaTex-document, refactoring of formulations and requirements ID’s.

Kristoffer Hilmersson

Customer survey. Generated Data models (E/R, Data Dictionary, Virtual Windows) and Quper in reqT with Thomas.

Marcus Hilliges

Elicitation techniques, design-mockups, result summary, general work on the system requirement document and parts of the validation checklist with Daniel.

Thomas Strahl

Data Dictionary, virtual window, Quper, overview of requirements and checking so they cover the product.