Danish Travel Card(Rejsekort) Proposal

Jorge Y. Castillo Rodriguez
Hussein Salem
Aku Nour Shirazi Valta
Faisal Jarkass
Dhiraj Bikram Malla
Xiaolong Tang

IT University Copenhagen
http://jycr753.github.io/SSEQ-Report/

November 18, 2013

Collaboration Protocol

Purpose: Shared agreements on overall forms of team collaboration.

Team members:

Hussein Salem

Faisal Jarkass

Aku Nour Shirazi Valta

Dhiraj Bikram Malla

Xiaolong Tang

Jorge Castillo Rodríguez

1. **What are the time slots that you can set aside for the team work?** Wednesday: after lectures till 14:00 [12:30-14:30] At a later point we might change the time We could work longer time in the future if we need to.

2. **If you disagree with each other or think that the solutions/products of some of the others is not satisfactory, what is the tone of communication?** Group discussion / democracy

3. **What is the maximum length of postings?** 500 words

4. **What do you do if a team member remains silent?** One of the team members will talk with that individual. And we might talk about it with the TA

5. **Do you expect that team members ask for help if they recognize, they do not manage their tasks?** Yes. We should have an open dialog where our problems and concern can be discussed

6. **Which team conflicts might arise and how do you imagine dealing with them?** Deadline issues, help the one that couldn't do it or perhaps give him more time.

7. **How and where do you share files?***You might consider google docs, dropbox, or the BSCW.* Google drive - File Sharing Google docs - Document editor Github - Latex Repository share

8. **Which tools for communication will you use for what purpose?** The main communication tool in our group will be Skype, since it is a free services that provides live chat, video and file transfer protocol if needed.

9. **how are you going to communicate asynchronously?** We decided to communicate through Skype and emails.

Prioritised List Of Software Qualities

Software Qualities	Points	Meaning
Safety	82	N/A
Security	**590**	Strong code structure in case hacking into system(F.x SQL injection) -Unit test -Errors report from users
Reliability	**560**	Computer program's ability to perform its intended functions and operations in a system's environment, without experiencing failure (system crash). IEEE-Std-729-1991: "Software reliability is defined as the probability of failure-free operation for a specified period of time in a specified environment" -Random testing -Redeveloping -Testing in different environments
Resilience	37	N/A
Robustness	210	N/A
Under-stability	210	N/A
Testability	270	N/A
Adaptability	310	N/A
Modularity	85	N/A
Complexity	120	N/A
Portability	50	N/A
Usability	**580**	The quality of user experience across websites, software, products, and environments. It is easy for users to understand and using the system. -Usability Design(UI) -Feedback from users
Reusability	82	N/A
Efficiency	150	N/A
Learnability	280	N/A

Development Model

In this section we will arguments the points for the chosen development methodology and how it can use to implement to develop the *Danish Travel Card*'s software while developing its software.

There is many developing software methodologies which describes the development life cycles and each its own importance in software development.

Waterfall Methodology

The **Waterfall Method** is one of the oldest and basic developments methodologies which still has a great influence in modern day software development and is follow by many software developments companies around the globe to organize their projects.

The Waterfall Method is a linear process where a sequential methodology is followed, progress is monitored and measured according to the completion of each phase.
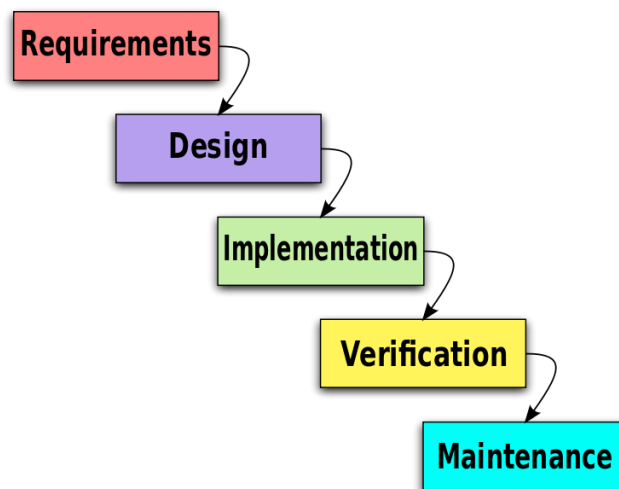

Figure 0.1: Waterfall Method

Waterfall Phases:

—**Requirements:** The first step in this methodology to start with requirement analysis and revise in the project is actually feasible with this methodology.

•**Design:** With the requirements and analysis acquire in the preview step, them a proper implementation strategy can be formulated according to the software environment. Further the design phase is divided in two sections. *system design and component design*. The system design contains details and specifications of the whole system and explains how each component of the system will interact with others. The component design contains specifications as to how each component will work separately.

•**Implementation:** In this phase is where actually some components start being created. The information collected in previews phases are applied in this step to create a functional software.

•**Testing:** This step is where the software developed in the preview phase is tested for any errors or discrepancies. In the Waterfall method the testing normally starts when the code is finished.

•**Installation:** After the software is tested, it can follow to the next phase where it gets installed in the computer or devices that it is build for.

•**Maintenance:** This final phase it could be stretched it for a few month or some years. With this follow up software bugs are fixed and adding/changing additional requirements or functionality to the software.

Advantages and Disadvantages

Advantages:

Since the waterfall method is the oldest and most widely used among software development companies there some advantages to this model:

1. Since it is a linear model, is very simple to implement.

2. The amount of resources to implement it are minimal.

3. Each stage is assigned to separate teams, this approach ensure project deadlines control.

4. Since the documentation is produced at every stage of the software development, this makes its understandably simpler.

5. After every major phase of software programming, testing is done tp the correct runtime.

6. Defined start and end points for each phase of the process, making it more easy to measured.

Disadvantages:

1. Fixed requirements difficult to change.

2. No final user visibility of the software until the development phase has been completed.

3. Clients are not very clear of what they want exactly from the software, when changes are mention in between it may cause a lot confusion.

4. Small changes or errors that may come up in the final software could cause a lot problems.

The waterfall works really well for commercial arrangements, but when working for internal customers its more difficult to implement.

Usage

TO BE DONE HERE

Document Convention

This document describes our convention on every document developed by the group or individuals. Which means that we will make sure that each document that we will produce is going to contain the following conventions (metadata):

1. ID
    a) Unique identifier consisting of integers

2. Name of task
    a) Title of the document

3. Version
    a) Version of the document, which will be added for each time the document is updated

4. Author
    a) The author of the document

5. Responsible
    a) The person(s) responsible for the particular document

6. Status
    a) The state of the document - done/undone/redo

Estimation and Planning

Planning Poker Estimates

Planning Poker, also call **Scrum poker**, using this planning model members of the group make estimates by playing numbered cards face down o the table, instead of speaking them loudly, later the cards get reveled and the estimates can be discussed[1].

Problem Domain Analysis

N0	Task	Winning Estimation
1	Explore the user content 1. Brainstorm on what research is necessary for your case 2. Implement the research 3. Document the results in an adequate manner 4. Refine your documentation	Hours 1. 1 2. 2 3. 1 4. 1/2
2	Rich pictures 1. Develop two Rich Pictures 2. Present the Rich Pictures for the whole group 3. Use the notes to write a short explanation of the rich pictures	Hours 1. 1 1/2 2. We were only "1" group 3. 1/2
3	Analysis class diagram over problem domain 1. Develop a class-event table about the problem domain of your case 2. Develop a class diagram for the problem domain of your case 3. Evaluate the class diagram 4. Describe the classes	Hours 1. 1 2. 11/2 3. 1/2 4. 1/2
4	Project Glossary 1. Create a glossary 2. Define the central terms in your analysis	Hours 1. 1/2 2. 1/2
5	Supervisor meeting	1/2 hour

---

[1]http://en.wikipedia.org/wiki/Planning$_{p}$oker

Our estimation was done in the group according to the principles of Planning Poker Estimation. We all had squared papers with different numbers within a range, which were representing a certain amount of time. For each task we laid down a random paper on a table facing down.

This process was repeated for each task until we were satisfied with result of the times given to a task.

Three Point Estimation

Quality in Use

N0	Task	Best Case	Most likely	Worst Case	Estimation
1	Use case diagrams and scenarios 1. Brainstorm on the tasks and activities the software is supposed to support 2. Development of scenarios 3. Brainstorm on the actors involved in the tasks 4. Actor descriptions 5. Use case diagram	Hours 1. 1 2. 1/2 3. 1/2 4. 1/2 5. 2	Hours 1. 1 1/2 2. 1 3. 1/2 4. 1/2 5. 2 1/2	Hours 1. 2 2. 1 1/2 3. 1/2 4. 1 5. 3	Hours 1. 1 1/2 2. 1 3. 1/2 4. 35 min 5. 2 1/2
2	Interface design using Mock-ups 1. Decide on 2 scenarios that you would like to work with in your mock-up workshop 2. Develop an initial interface design 3. Prepare 2 mock-up workshops 4. Implement the 2 mock-up workshops 5. Document the results	Hours 1. 1/2 2. 1/2 3. 1/2 4. 1/2 5. 1/2	Hours 1. 1/2 2. 1 3. 1 1/2 4. 2 5. 1/2	Hours 1. 1/2 2. 1 1/2 3. 2 4. 2 1/2 5. 1	Hours 1. 1/2 2. 1 3. 1:25H 4. 1:50H 5. 0:35H
3	Revising project team protocal 1. Changes to the Project Team Protocol 2. Propose and discuss changes in the project team	Hours 1. 1/2 2. 1/2	Hours 1. 1/2 2. 1/2	Hours 1. 1/2 2. 1/2	Hours 1. 1/2 2. 1/2
4	Supervisor meeting	1/2 hour	1/2 hour	1/2 hour	1/2 hour

$Estimation = BestCase + 4XMostLikely + WorstCase$

In this case we calculated the estimated time for each task based on the three cases; *best case*, *most likely*, and *worst case*. Each case was given a time which we as a group discussed and agreed upon. After all three cases were set, the estimation was calculated based on the estimation formula seen under the table $BestCase + 4xMostLikely + WorstCase/6$. The procedure was repeated for all the tasks.

Risk Analysis

ID	Description	Probability	Effects	Strategy
1	Group members leave the group	High	Documents and responsibilities is left undone, this may lead to delays in the project	The rest members will have to develop more, this will mean those responsibilities left behind will be divided with in the rest of group.
2	Time fixing/improving the software after debugging	High	More work have to be done	Discuss the case in details and use the supervisor
3	Document delays	High	That the specific group member can't take new tasks	Project manager has to follow up more regularly
4	Planning and Estimating	High	The plans and estimating document has to done again	Use the supervisor to acknowledge the understanding and discuss in detail in the group
5	Misunderstanding of the task	Moderate	The developed tasks is done wrong	Discuss every tasks in the group and/or use the coffee hours to understand
6	A student becomes ill	Moderate	Can't participate in supervisor meeting or can't help developing some documents	Work in groups of two at least or help him come back on track with his work
7	Underestimate size of task	Low	More work have to be done	Discuss the case in details and use the supervisor
8	Tools for debugging	Low	Product and Project delays	Use the supervisor and make a lot of research about the tools
9	Hardware failure	Very low	Losing some documents	Backup more frequently and all group member has a copy
10	Inexperienced	Very low	The documents and/or debugging of the system isn't done properly	Use the supervisor, teacher and make a lot of research

Probability:
Very low < 10%
Low 10% - 25%
Moderate 25% - 50%
High 50% - 75%
Very high > 75%

Project And Product Risks

Risk	Affects	Description
Staff turnover	Project	A student gets sick or leaves the group
Management change	Project	The role project manager is going to rotate, could lead to changes in priorities
Hardware failure	Project	Hardware fails before taking backup
Debugging change	Project and Product	There will be a larger number of bugs found than first anticipated
Size underestimate	Project and Product	The product to debug is bigger than first anticipated
Planning and Estimating	Project and Product	Some tasks take longer time than first anticipated
Document delays	Project	Some group members doesn't deliver a document at specific deadline
Tools for debugging	Product	Tools to debug doesn't seems to be good as the provider says
Inexperienced	Project and Product	A group member can't deliver a high quality of document or misunderstand the task
Inexperienced Project manager	Project and Product	A group member as a Project manager doesn't priorities correct
Poor Project planning	Project	Dependencies of other documents/tasks leads to delays

Term	Definition
Casual user	A person that uses the rejsekort 1-4 times a week.
Laggers (user)	Persons that are slowly to adapt to new technologies.
GPS	Global Positioning System.
Use Case	A list of steps that defines the interaction between the user and the system.
Roles and Responsibilities	A group members role and the responsibilities in the project for the specified date.
Software Qualities	The qualities that are important and agreed upon in the group, which is the corner-stone of the development model.
Work Breakdown Structure	A way to structure the complete project work.
Risk Analysis	Analyze the risks in the project.
Use Context	The context in which the software is deployed.
Rich Pictures	Represent relevant aspects of the problem and application domain as rich pictures.
Interface design	The design of the interface that the user sees.
Mock-Up	Is a scale or full-size model of a design or device, used for teaching, demonstration, design evaluation and promotion.

Interviews Summary

Interview to ticket controller

26. sep. 2013 15:05 at DR Byen station

   Have the Rejsekort system made your work easier? *"In the start there were many problems, but it have become better over time, which makes by work a bit easier"*

   Which problems do you see the passengers face? *"The passengers using travel card(Rejsekort) forget to check in again when they change transport means under their commute. This entitles them to a fine, which we (controllers in the metro) have stopped to hand out, as there have been much confusion since the travel card came"*

Interview to IT professionals

Observation *26/9-2013 15:00-15:20* at DR Byen station A guy is coming up to the station. He uses his clips card and walk away. A metro just comes in to the station. Around 8 passengers come out. And 2 people get into the train. A little girl comes and check out with a Rejsekort. Then a guy is coming up to the station. He uses his clipe card and wait for the metro. Two guys are coming to station and use their clipe card. After that they are waiting for the train.Then a man put his wallet close to check in machine. Then a man and a woman are coming up to DR station. The man check in with his clipe card. the woman put her wallet close to the check in machine. A guy comes up with his Rejsekort. He checks in and wait for the metro. Then a train is coming into station. A woman is very hurry up to use her clipe card. But her card is bended. So she tries many times and finally she check in with her clipe card. But she missed the trains.

Another train is coming in to station. Many passengers are coming out. Passengers who use rejsekort are standing around the check out machine. Because they are waiting for the passengers who are in front them to check out. Two little girls, they are coming up to the station. They both check in with rejsekort. And they are waiting in the train station. A guy is checking out, but have problems seeing the display on the machine because the sun is reflecting on it. Another woman is checking several people in, but seems a bit confused.

27/9-2013 8:30-8:45 DR Byen station A guy who take a cup of coffee is coming up to the train station.He puts his rejsekort from his pocket. And he puts close to check in machine. The trains are coming every 4 minutes. There are a lot of people are waiting in the train station. There are crowd of people are standing in front of check out machines. Then a train is coming to station. there are around 12 people getting off from that train. A girl takes rejsekort. She press the machine. She checks in not only for her but also for her friend. There are still a crowd of people in the station. There are 3 people standing in front of check out machine.

A train is coming in to station. Most of people go inside. And two little girls are coming out of train. Two men are coming up to station. They both check in with rejsekort.
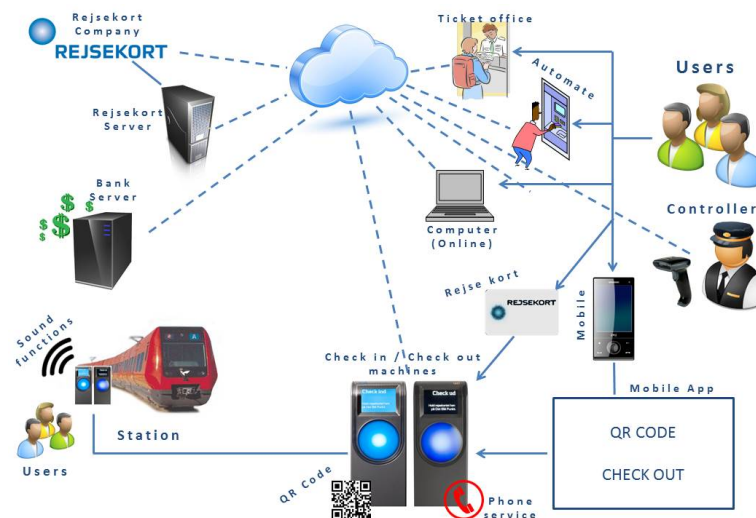
Rich Picture Diagram

Figure 0.2: Class Diagram Draft

Rich picture description

The rich picture basically illustrates our intended mechanisms in the Rejse kort system improvements. The illustrations is made up of the basic stakeholders, users and devices, and how they are connected.

Used notations

Figure 0.3: Rich Picture Diagram

The striped lines shows that a certain stakeholder, user or device is connected to the *cloud*, which is connected to Rejse korts server. The arrowed lines shows what a user can use in the system. The simple line is used to further illustrate what the components contains or are used for.

Class Diagram

Description

User

There are two types of users in this system. They are customers who use the system to travel. And also employee works in the public transport company. And both users can access to users' status(check in/out) and location (zone area).

Customer

Customers have more functions than the User class. Customer has the balance value. That means customers can check and charge their own balance accounts. The customers have two ways to use the system. One is rejsekort. Another is mobile app.
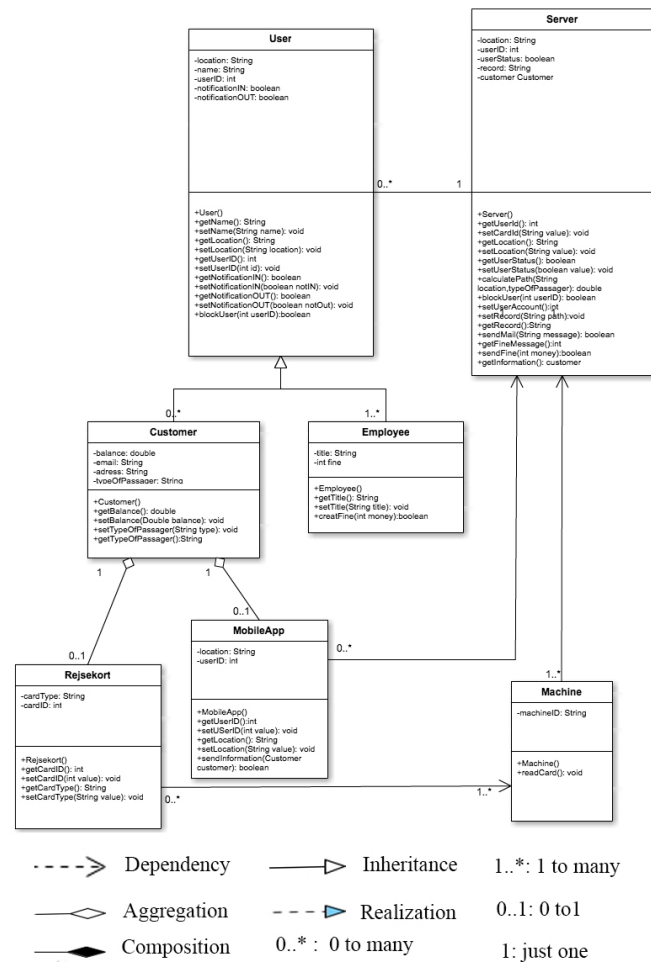
Figure 0.4: Class Diagram

Employee

The employees who use the system are ticket controllers. They can only access to users' ID, status and location.

MobileApp

Mobile app can get users' ID, status, location(with GPS) and return all the data back to the server. Users also get balance data from server.

Rejsekort

Customers can always use the card to check in/out and check balance through the machines in the station. The card contains customers' ID, name and etc.

Machine

Machines are located all the stations and buses in Denmark. The machines can read rejsekort. And they can send customers' informations and location back to server. And the machines can also get customers' balance base on customers' ID from the the server.

Server

The server contains all the customers' ID, status, location, balance and customers' information. The server can receive all the data from the mobile and the machines. The server can also calculate the price base on check in/ out location and charge that price from balance account.

Use Case Description

Actors

Customer: Costumers is all kind of persons that does use *Rejsekort* system to travel from a location to another, this include regular travelers, such people that goes and back from work, kinds/young that goes to school, disables, retired people. So technically an user is all costumers that uses the system.

Customer service: Technically this a person who works for the Rejsekort's company as part of their customers service stuff. This person does have all the rights and permission to control all users account such, balance check, history, block card. But it can not control the actions that require physical response from the user such *Check in* and *Check out* since this only can be done by the user true a rejsekort machine or smart phone application.

Controller: The controller is the person who checks if a passenger is traveling with a properly validated card or proper checking with the mobile application. Otherwise the controller has the power to issue a fine to the infractor.

Financial institution: Financial institutions can process the request from users to add credit to their rejsekort from either a credit card, Dankort or direct transfer, but this function can be trigger as well true costumer service.

Use cases

BlockAccount: An account can be blocked either by the customer or the customer service.

CheckBalance: This function in the software is to allow users to check their own account balance or either call costumer service and check it true them.

BlockCardAccount: A customer or the customer service can block a specific card on a customer account.

CreateAccount: A customer can create an account or request the customer service to create it.

AddBalance extends Withdraw : User can add credit to its balance or contact consumer service to do it.

CheckOut extends Withdraw: When a passenger ends his commute he is able to checkout, which also can be done by the customer service in case it is needed

Withdraw: Depending of customer's choice, the financial institution withdraws money from the customers bank account, credit card or Dankort, then the balance is added to the Rejsekort account credit, also this function can be trigger if auto fill up is enable; this means the balance will auto fill up with a pre conditioned amount if the balance is too low after a commute has be done.

CheckHistory: The customer service or the customer himself can check the history of activities on the account.

CheckCustomerStatus: It is possible for the customer, the customer service and the controller to check whether the passenger traveling on Rejsekort have been checked in.

SendMail: Basic email communication format via email from users to costumer service and vice versa.

EraseFine: The customer service can erase a fine given to a customer.

CheckIn: Costumers traveling can check in, also they also have more options while check in, such as:

1. AddBicycle

2. AddAnimal

3. AddAdult

4. AddKid

GiveFine: The controller can give a passenger a fine.

Configuration Management Plan

Objectives and scope

Even though there is no way to manage what it is unknown, it is essential to have a detailed knowledge of the of the infrastructure in order to make the best use of it, to develop a good CM it should include among other things:

- Faster problem solution, for better quality of the software and the service, a common problem is the connection to the servers may be down, errors in the magnetic reader of the card/login machine or is not up to date with the latest software containing the latest encryptions keys. Having to detect this most common errors without having a optimal configuration management database can increment the solving time of the problem in a larger scale.

- Efficient change management, it is very important to know all the changes made to the system or structure of it, this way changes can be design not to produce errors of incompatibilities or problems.

- Cost reduction, detailed documentation of all the elements of the configuration allows to avoid unnecessary duplicates, for example: a) Cost of licenses by avoiding having multiple software that has the same functionality b) Illegal copies could carry viruses or any other kind of malware that could put in risk the software/hardware in risk c)Legal requirements with licenses or any other third party application legal requirements that could have a negative impact in the organization and by extension in the system.

- Security it is a big part of our software and having its configuration management database up to date allows to find and fix vulnerabilities in the software.

Terminology

To have a proper CM there is the necessity to have some pre-defined rules, which includes the follow:

- Naming conventions this agreement will be a contract to keep the same name structure for all files, folders, classes, methods, for example:
  - **Classes** it will always start with a package name abbreviation, capital letter for each different word describing the class for example: $VYXZSomeReallyImportantClass\{\}$
  - **Method**, it will have same format as it predecessor but no including the package name, but it will also include what type of method it is or what kind of value it does return or print in the software scope, for example: $SomeImportantMethodBoolean()\{\}$
  - **Folders** and txt files this will always need to have the $date(YYYYMMDD)$ that was created, person initials, a location of the original file, every different values separated by a dash ($-$), for example: $20131023 - GD - JC - filename.extension$
  - **Version** number of the software it will go with 1.0 for the start, if there is bugs fix it will increment by 0.1 and if their major changes to the system it will increment by 1.0
  - **Source** control by using git to control the status of the system, by dividing the source into different branches and only margin them into the final products when all steps of debugging and cleaning the code has been done.
  - **Store location** to keep control of all files that are produced during project evolution, most of files are stored in Google Drive for backups, and the rest of "Software" related files are placed into a private repository to make it accessible from different locations to programmers, developers and designers of the system.

Collaborators

During this project since we are using the initials of the person to define the file that it is being created so each member should have an unique initials containing 2 - 3 character depending on availability and a git or mercurial username to being able keep control of our document changes in the central repository or own branch.

List of collaborators:

1. Jorge Y. Castillo (*JC*) - *jycr753*

2. Xiaolong Tang (*XL*) - *xiaolongtang89*

3. Faisal Jarkass (*FJ*) - *faisal1985*

4. Dhiraj Bikram Malla (*DB*) - *djbik*

5. Hussein Salem (*HS*) - *husalem*

6. Aku Nour Shirazi Valta (*ANV*) - *nourshirazi*

7. All members (*AM*) - N/A

This guidelines will help to build a solid CM that it is required to have a efficient software development environment, and post software maintenance or bug fixing. Even though this rules are important and be follow, there is always exceptions for example: Initial software structure mock ups, or hardware placement mock ups.

Control of sub responsibilities and 3rd party software or hardware contractors, to control what the 3rd party software developers have access the only thing that it is available to them, is close source API with the correct list of classes that they can use and how they can extend to be able to do their job programming the required extension for the software.

Software Requirements Specification

This **SRS** needs to have some basic structure that will made its use more efficiently when following its guideline, this have to be seeing as a guideline more than an absolute guide, never the less following this document will void major flows and errors in the software while developing the project or afterwards. In this case the basic document architecture consist in the follow:

⇒ **Introduction**: Describe how requirements are handled for waterfall model and the system's function.

⇒ **Requirements**: Describe the functional system requirements and non-functional requirements.

⇒ **Glossary**: Define the technical terms used in this document.

Introduction

In software development, Requirements phase focuses on defining and capturing the needs and problems that a software application is to address and solve. The requirements phase is the first phase in waterfall development model, setting the stage for the rest of the phases of the software application development.

The requirements phase aims to come up with Requirement Specification document or documents. The aim is to define the requirements in as clear and as detail manner as possible. Normally in order to capture, collect and gather the requirements; a dedicated team is setup to capture the requirements.

Requirement

In our software requirement document we are going to use the graphical notation and structured specifications, in order to handle and define each requirement. The choice of the graphical notation is based primarily on the fact that it is far more readable for us as developers, the structured language notations use templates to specify system requirements.

Software description:

Requirement ID: Identification number for specific requirement **Title:** the title of the specific requirement **Description:** Specific description and details on what the requirement functions does, and how to how to proceed when an action takes place **Priority:** Define the priority of this requirement for the entire project **Risk:** Define the risk if this requirement is not meet.

Priorities:

Low: Good to have functionality but not required to make the software work. **Medium:** Some functionality that could have to implement the UI/UX while interacting inside the GUI **High:** It must have to function

Risk:

Critical (*C*)**:** It must be developed to make the software work **Medium** (*M*)**:** Optional functionality but not essential to make the basic software work **Low** (*L*)**:** Function that will be implemented for as a complement for the software but present no risk for its functionality.

An adult is a person in the age of 16+, where a kid is a person in the age 12 − 16. For kids under the age of 12 the journey is free of charge, but the person has to be accomplished by an adult or he has to pay the minimum fee.

Requirement ID	01
Title	Block Account
Description	User can block his account that which put a stop to any kind of travel or transactions made with the travel card or mobile application until further notice.
Priority	High
Risk	Critical

Requirement ID	02
Title	Check Account Balance
Description	This required the such account has been authenticated, and show the account current balance.
Priority	High
Risk	Critical

Requirement ID	03
Title	Create Account
Description	Open an account, which it has to include the username, last name, billing address, telephone, and credit card information. This has to be made true a secure channel (https) with their valid SSL certificate and have a two steps verification to validate users fx: SMS to verify phone.
Priority	High
Risk	Critical

Requirement ID	04
Title	Check History
Description	This required to have access to the users travel data, balance history and spending data, to be able to show a complete list or historical spending of the users.
Priority	High
Risk	Medium

Requirement ID	05
Title	Check Customer Status
Description	Make possible for: customer, customer service and controller to check whether the passenger traveling on Rejsekort have checked in.
Priority	High
Risk	Critical

Requirement ID	06
Title	SendMail
Description	Both the customer and service can communicate by mail
Priority	Medium
Risk	Medium

Requirement ID	07
Title	EraseFine
Description	The customer service can erase a fine given to a customer.
Priority	Medium
Risk	Medium

Requirement ID	08
Title	CheckIn
Description	A passenger traveling with Rejsekort can check-in when he starts his commute and if he wish he can add additional passengers such as adults and kids, bicycles or animals
Priority	High
Risk	Critical

Requirement ID	09
Title	GiveFine
Description	The controller can give a passenger a fine.
Priority	High
Risk	Critical

Requirement ID	10
Title	BlockCardAccount
Description	A customer or customer service are able to block a specific card
Priority	High
Risk	Critical

Requirement ID	11
Title	AddBalance
Description	The customer service or the customer can add balance to his account.
Priority	High
Risk	Critical

Requirement ID	12
Title	CheckOut
Description	When a passenger ends his commute he is able to checkout, which also can be done by the customer service in cases it is needed.
Priority	High
Risk	Critical

Requirement ID	13
Title	AddBicycle
Description	A passenger traveling with Rejsekort can check-in when he starts his commute and if he wish he can add an bicycle to his journey. It is permitted only with two wheel bicycle.
Priority	High
Risk	Medium

Requirement ID	14
Title	AddAnimal
Description	A passenger traveling with Rejsekort can check-in when he starts his commute and if he wish he can add an animal to his journey. For animals meant Small dogs, cats, birds and other small animals. Animals may be included for free, but must be placed in a bag or cage and remain there for the whole journey. For larger dogs mean dogs that can not be carried in a purse, a cage or similar. Each passenger may only include one large dog, and this must be on a leash and under the passenger's control. For dog paid child price.
Priority	High
Risk	Medium

Requirement ID	15
Title	Add Adults
Description	A passenger traveling with Rejsekort can check in when he starts his commute and if he wish he can add an adult to his journey.
Priority	High
Risk	Medium

Requirement ID	16
Title	Add Kids
Description	A passenger traveling with Rejsekort can check in when he starts his commute and if he wish he can add an animal to his journey.
Priority	High
Risk	Medium

Requirement ID	17
Title	Withdraw
Description	Depending on the customers choice, the financial institute withdraws money from the customers bank account when balance is added to the Rejsekort account or when checking out.
Priority	High
Risk	Critical

Glossary

A functional requirement has the following properties:

| Requirement ID	Uniquely identifies requirement
Title	Gives the requirement a symbolic name
Description	The definition of the requirement
Priority	Defines the order in which requirements should be implemented. Priorities are assigned values from low to high
Risk	Specifies the risk if this specific requirement is not implemented. It shows how critical the requirements is to the system as a whole. The following risk levels are defined over the impact of not being implemented correctly. -Critical (C) it will break the main functionality of the system. The system cannot be used if this requirement is not implemented. -High (H) It will impact the main functionality of the system. Some function of the system could be inaccessible, but the system can be generally used.
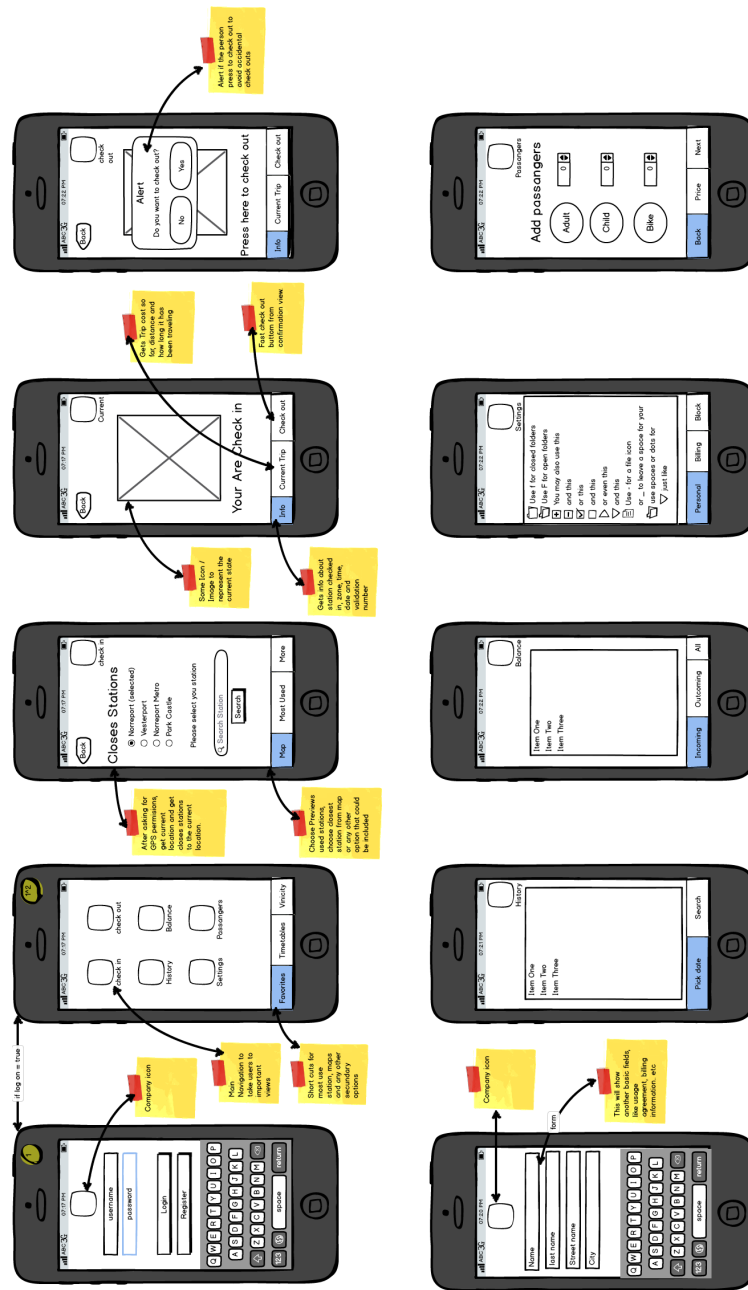
Figure 0.5: Mobile App Mock op

Quality Assurance Plan

First we need to define what *QE(Qualities Ensures)* roll will be in our software project, to understand this there is need to be defined, this method is widely used in manufacturing industries, but its methods and terms can widely applied in software developments. When applying this methodology it defines the standards and quality of the final software to insure that the final product becomes a high quality product.
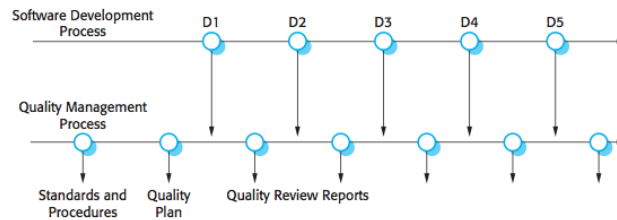

Figure 0.6: Chapter 24 page 653

Quality management ensures there is a check for quality of the software during each phase of its development, lets take into consideration the above graphic: Lest take $D(x)$ as each milestone in the waterfall method, and Q as the quality management process, so in other words for each D that is complete there will be need to check D and its quality requirements before proceeding to the next development phase.

By producing an accurate and well defined quality management document that includes the desire software qualities and describes how this are assessed.

According to Humprey (1989)[2], he suggest a solid structure for the quality plan, which includes:

1. **Product introduction:** A description of the product, its intended market, and the quality expectations for the product.

2. **Product plans:**The critical release dates and responsibilities for the product, along with plans for distribution and product servicing.

3. **Process descriptions:** The development and service processes and standards that should be used for product development and management.

4. **Quality goals:** The quality goals and plans for the product, including an identification and justification of critical product quality attributes.

5. **Risks and risk management:** The key risks that might affect product quality and the actions to be taken to address these risks.

Software testing is an essential and important part of the waterfall model, which also is the main part of the quality assurance activities. In the waterfall model there may be phases which does not explicit explains the quality assurance activities, but an assurance can be included where needed in the shifts of the different phases, such as the inspections and the reviews etc..

Most of the activities in our quality assurance plan can be summarized in the model seen at left. The model illustrates the different quality assurance activities related to the waterfall model **??**[3] and its processes. If we compare Humpreys (1989)[4] suggested points to the quality assurance in the waterfall model, they can be summarized to three important key characteristics:
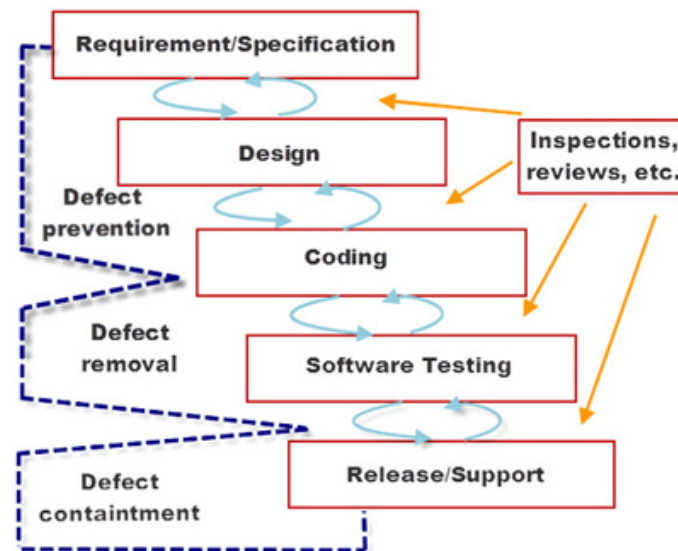
1. The testing phase, which is the main stage of the quality assurance activity.

2. Inspections and reviews that are made and carried out at the transactions of the phases.

3. Preventing errors especially in the early phases, and removing them during coding and testing.

These three key point have to be implemented into the Rejse Kort project in order to make sure that the qualities are meet and lives up to what is expected. The implementation of these quality assurance activities related to the Rejse Kort system are explained below.

---

[2]Software Engineering page 653b
[3]http://qatestlab.com/knowledge-center/QA-Testing-Materials/waterfall-process-in-the-quality-assurance-activities/
[4]Software Engineering page 653b

Testing phase

In our case, we design the unit test before we start coding. After coding, the program return the results that should pass the test code.

We also going to test the functionality, in order to do so we will use black/white box testing.

The black-box test, tests the functionality outside the software, ex. the application developed for the customers, has some intended functionalities, that we will have to ensure works properly, and doesn't act crazy. Here we will have to create a document which specifies each test and document the outcome. On way of doing it, is to use fabricated answers, which means that we know what the result will be and then compare it against the outcome of the application.

The white-box test, tests the functionality inside the software, which means that each method, class etc. has to be tested, in order to achieve it we will have to provoke the application and see the outcome. Here we will also have to produce a document, with the specified tests, and compare it against the fabricated answers.

Inspections and review

The inspection and reviews should take place at every transaction of a phase is order to check whether the output lives up the intended result. The quality reviews will be based on documentation that are produced during the different phases, and necessarily at the software development processes. The reviews will check the consistency and completeness of what is produced during the phases in the waterfall model.

It is worth mentioning that the reviews and inspections should be tools to improve the Rejse Kort system, and not be used as tools to inspect or access the performance of the teams or the individual who are part of the process.

Our review will basically be divided into three steps in every phase:

1. Pre-review activities

2. Review meeting

3. Post review activities

The pre-reviews activities focuses on the review planning and preparation. Once these pre-reviews activities have been settled a review meeting can take place where the authors of a document or other *"products"* being reviewed should *"walk through"* or explain the *"product"* with those whom are reviewing it. Once a reviewing meeting have taken place, the post review activities can take place, which handles and should address the problems and issues which were addressed at the meeting.

Preventing and correcting error

The program must pass all the test code before we use it. If the program cannot pass the test code (errors and receptions), we will use log to record it, trace the errors/receptions and fix the code. After updating the code, we test the program again and again until it can pass the test code.

Conclusion

This course has been very enriching and enjoyable. It has tremendously contributed to our growth as master/bachelor students. Every lecture and document developed in this course has been a challenge, a challenge that we weren't quite use to handle. Because develop documents was a new experience in our career which forced us to think in a differently. While developing and idea to find the proper solutions to a given problem was also new to us. Which forced us to think in a more abstract way and fully understand the documents, models and methods to develop this ideas.

Before the course we thought this tasks could have been quite easy to achieve, but during the course it became very obvious that we had to put a lot of effort, in order to keep up doing and understanding the objectives.

The course was very well-designed for each activity added to the larger objective of the course. And, we believe the course succeeded in achieving the goal that was set up for us.

Which also meant that we had to argue, for each choice that we made, in order to justify the choice. And not chose based on what we think is *IN*. Like in the choice of the development model, one could argue to chose scrum, cause that is what are popular today (in terms of widespread use in the industry), but ain't the best choice for our project, because our project description ain't going to change during the development, but rather fixed. So our choice fell on the waterfall-model as it best fits our project.

The exciting practical component of the course, coupled with strong theoretical foundations, gave us opportunities to explore and experiment with different models in software engineering.

We have learned to critically examine the choices we made, and to reflect on how well our models align with the experts in the field.Overall, after finishing this course, I feel that I gained a variety of technical skills that will be helpful in other classes and in our future career. In addition, all the feedback that We got from others will help us focus on the areas that We need to work on. For this reason, We are glad that the final project was to do this e-portfolio because it allowed us to revise and reflect on everything that We have done for this course.

After We re-read and revised all of our work, We have to admit that seeing the work that We had completed for this class amazed us. Before the course, it seemed that the workload was a lot, but now We understand how helpful it was to put most of my ideas in written assignments.

Roles and Responsibilities

Hussein Salem	Aku Nour Shirazi Valta	Dhiraj Bikram Malla	Faisal Jarkass	Xiaolong Tang	Jorge Rodriguez
Role: date:25/9/13 Configuration manager	Role: date:25/9/13 Preparation	Role: date:25/9/13 Maintenance	Role: date:25/9/13 Preparation	Role: date:25/9/13 Project leader	Role: date:25/9/13 Executive assistant(minutes)
date:2/10/13 Maintenance	date:2/10/13 Configuration manager	date:2/10/13 Preparation	date:2/10/13 Executive assistant(minutes)	date:2/10/13 Preparation	date:2/10/13 Project leader
date:9/10/13 Preparation	date:9/10/13 Executive assistant(minutes)	date:9/10/13 Configuration manager	date:9/10/13 Project leader	date:9/10/13 Configuration manager	date:9/10/13 Preparation
date:23/10/13 Executive assistant(minutes)	date:23/10/13 Configuration manager	date:23/10/13 Project leader	date:23/10/13 Preparation	date:23/10/13 Preparation	date:23/10/13 Configuration manager
date:30/10/13 Configuration manager	date:30/10/13 Project leader	date:30/10/13 Configuration manager	date:30/10/13 Configuration manager	date:30/10/13 Executive assistant(minutes)	date:30/10/13 Preparation
date:6/11/13 Project leader	date:6/11/13 Configuration manager	date:6/11/13 Preparation	date:6/11/13 Prepation	date:6/11/13 Configuration manager	date:6/11/13 Executive assistant(minutes)
date: 13/11/13 Preparation	date: 13/11/13 Preparation	date:3/11/13 Executive assistant(minutes)	date:13/11/13 Configuration manager	date: 13/11/13 Project leader	date: 13/11/13 Configuration manager

Project Leader

Leads the project for the particular week, which means that we chose a project leader each wednesday. And for that particular week, he is in charge of the work flow, and distributes the rest of the responsibilities to the rest of the team members. In addition he is the one controlling the group discussions. The project leader cannot overrule an earlier decision except if the majority of the group wishes to.

Preparation

Preparation of supervisor meeting. Which mean that he is the one responsible for the meetings success. In order to get a productive meeting, the one in charge of this role has to prepare some questions for the TA. That means that he has to interact with all group members in order to collect all the doubts. This will lead to the agenda of the meeting.

Maintenance

The maintenance responsible has to maintain the google drive, which means that he knows where the documents are placed and that they are placed in the correct directory.

Executive assistant

The executive assistant takes minutes of the supervisor meeting and minutes of the group meetings, in order to collect all the information being discussed. Prepare research documents for the particular week.

Configuration manager

The configuration manager is responsible of the consistency of the developed documents. So that the changes are handled systematically. And keep track of the changes that are being made in the project.

Reflection on our Project Team Protocol

Did you follow the agreement?

We did follow the agreements seen from an overall perspective, but at times changes an comprises were needed.We agreed to meet as a group in order to work on the project at Wednesdays after calls, which we mostly did, but this quickly changed as we realized it wasn't enough. We kept meeting on Wednesdays, but had to meet during some of the other days as well. We did see this coming, which is why we in the protocol included that we at times would have to set some more time off. For the most of the time we managed to follow our time slots where all team members were present, but at other times this was harder to realize. In cases where some group members couldn't be present, the rest of gathered, and those who wasn't present could stay tuned via Skype.

We agreed to have decent group discussions in cases where we disagreed, and our result should be based on democracy. We did face disagreements at times, but as a group we managed to come with a solution and option that we could all agree upon.

Our group had a good working environment and a smooth process, which made all members comfortable and as a part of the group, and therefore all members contributed to the project and did not keep silent.

We agreed to have an open dialog in case any members did manage to make a task that were given them to them. Most of our work were made where all members were present, and we therefore faced and solved most of the problems together. In the cases where a single members faced problems, an open dialog took place where all members added their perspective on the problem and how we could solve it.

We expected to face conflict concerning deadlines how we could manage the time. As mention most of the project was done together in the group, and we therefore faced the conflict together. At times where we did not manage live up to the deadlines we had to schedule more time, which we mostly did, but at times this simply wasn't possible which made us divide the work and afterwards reviewed it together.

Most of our sharing were done using Google Drive and later on at GitHub aswel. In the early process of our project out Google Drive failed us where we lost most of our work. Luckily we had a back up which were restored. When we were not present and communication was needed it, we used Skype as the main tool and emails if it was more efficient in some cases.

Did the agreement fulfill the purpose?

Our intention related to the time slots we agreed upon, was to make sure all group members could contribute to the project and be a part of the whole process. Our agreement did fulfill the purpose since our intention was met.

The purpose of the upon and decent group discussions was to make sure we could come with a solution that we could all agree upon and be stratified. This purpose was mostly fulfilled, but at times some compromises had to be made.

In case any members faced some problem the purpose of the open dialog made sure that the members could feel open to share the problems and not keep them to himself. This was actually a very important agreement, which purpose was met, since in a case where a problem was not shared, it would have affected the whole project in the end.

The purpose of scheduling more time in case we did not live up to a certain deadline, was to make sure we would finish the part. The purpose was mostly met, but pushed the deadline of other parts further, which in the end resulted in a certain time pressure.

The tools we used for document sharing and communication made sure all group members was updated and could follow the changes that were made.

Would you, with your experience propose a different agreement?

Seen from an overall perspective we think the agreements were suitable for our group since most purposes were met. It could though in some cases have been efficient if the agreements had been described in more details. This could have given us a deeper clarification when issues arose and easier to find a solution while referring to the agreement.

What items would you recommend to include in the project team agreement

It would have been useful and very helpful, if we from the beginning had agreed on how the work structure should take place. An agreed work structure could have made us more efficient concerning our time management.

Agreeing upon which text editor we would use could likewise have been efficient, since we needed to convert documents back and forth. This could at times be very confusing and hard to keep track on the document version.

Seen from an overall perspective we satisfied with the outcome of the project, even though it was a learnful challenge. We managed to look into all the assignments and its related theory. The course, the related project and the process we undertook have been interesting and useful, especially when seen from a professional perspective related to the activities in software engineering. The course addressed a wide range of aspects which gave us a insight and understanding of the processes of software development takes place. Working with the Danish Rejse kort system reinforced or insight and understand, since it was something we could relate to in our daily life.

If we look in to each separate case, we think there a still areas which can be improved. When we reached a certain points in our project, we reflected on some of the work we had done earlier on, and could easily that we could made it a different way or improved it. We did manage to improve some of the work based on the reflection and also based on the reviews, but there is to a degree still room for improvements.

Dynamic Test Specification

In the Waterfall model we have chosen includes a testing in phase 4. After developers have developers are done with coding and provide final build to testers, testing starts in this phase. Through this phase we intend to reflect on the following points:

Test planning

In the test planning we will decide what we want to test, define the test process, schedule the testing, estimate testing effort and assign testing resources. We will further consider how to report the test result and how they are handled

Design of test cases

In our test cases we intend to include a unique identification, execution preconditions, data and actions inputs, our expected results and finally trace the individual tasks. These task will be done in a systematic manner.

Prepare Test Data

At this stage of the process we will prepare the data which we intend to test

Prepare Test Environment

At this stage of the process we intend to prepare the test environment. In the testing environment for our case, we are going to prepare a computer as the server. It supports to running all the time during the test. We also need a card reader machine and a mobile phone. They all have the internet accessed. On the server, it is running the server program and test program. The card reader machine and mobile should run the client program.

Run Program on Test Data

The test program can monitor the client side(card reader machines and mobile app) if connect to the server and the data return back to the server. And it also can monitor the input and output data on the server.

Compare Results to Test Cases

In our case, when we run the test program on the server, we will get the output as data. We see if the data as what we expected.
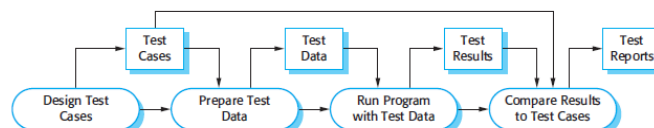
Report Testing

We put all the testing data as log that saves in the server.

Correct Errors

If the output data is not what we expected, we will find the errors and fix it.
    Our overall process can be summarized in the figure seen below, which is a traditional testing process.

Test Cases

Test Case 1: Check In - Card

1. ID
    a) TC1

2. Pre-con
    a) System ready for card input
    b) Connected to the server
    c) The card should be functional
    d) There is money on the card/account

3. Input
    a) Data from the card

4. Expected result
    a) Ok message indicating that the check-in was successful

Test Case 2: Check In - Mobile

1. ID
    a) TC2

2. Pre-con
    a) System ready for card input
    b) Connected to the server
    c) The mobile app should be functional
    d) There is money in the account

3. Input
    a) Data from the mobile app

4. Expected result
    a) Ok message indicating that the check-in was successful

Test Case 3: Check Out(card)

1. ID
    a) TC3

2. Pre-con
    a) System ready for card input
    b) Connected to the server
    c) The card should be functional

3. Input
    a) Data from the card

4. Expected result
    a) Ok message indicating that the check-in was successful

Test Case 4: Add Balance

1. ID
    a) TC4

2. Pre-con
    a) Connected to the server
    b) The account should be valid and functioning
    c) Internet must be accessible

3. Input
    a) Account transformation from the user's bank account to the travel card system

4. Expected result
   a) Updated user credit in the travel card system

Test Case 5: Check Customer Status

1. ID
    a) TC5

2. Pre-con
    a) Connected to the server
    b) The system should be functional

3. Input
    a) Users travel card or app

4. Expected result
    a) Status of whether the user is checked in or not

Tasks we have already performed

Activities	Done by	Date
Group members info	Group	4/9/13
Collaboration protocol	Group	10/9/13
Explore your case(abstract)	–	12/9/13
Kick-Off	Group	4/9/13
First TA meeting	Group	11/9/13
Development Models Overview	Group	13/9/13
Select a Development Model	Group	16/9/13
Work Breakdown Structure	Group	11/9/13
Explore your case(abstract)	Group	16/9/13
Select a Development Model	Group	16/913
Estimation and Planning	Group	16/9/13
Roles and Responsibilities	Group	16/9/13
Risk Analysis	Group	16/9/13
Explore the Use Context	Group	25/9/13
Rich pictures	Group	25/9/13
Analysis Class Diagram over Problem Domain	Group	25/9/13
Project Glossary	Group	25/9/13
Use Case Diagram and Scenarios	Group	1/10/13
Interface Design Using Mock-Ups	Group	1/10/13
Revising Our Project Team Protocol	Group	1/10/13
Document Convention	Group	8/10/13
Configuration Management Plan	Group	8/10/13
Document Project Requirements	Group	8/10/13
Quality Assurance Plan	Group	23/10/13
Dynamic Test Specification	Group	6/11/13
Software Architecture Sketch	Group	15/11/13
Reflect on Your Project Team Protocol	Group	15/11/13
Finish the project	Group	18/11/13
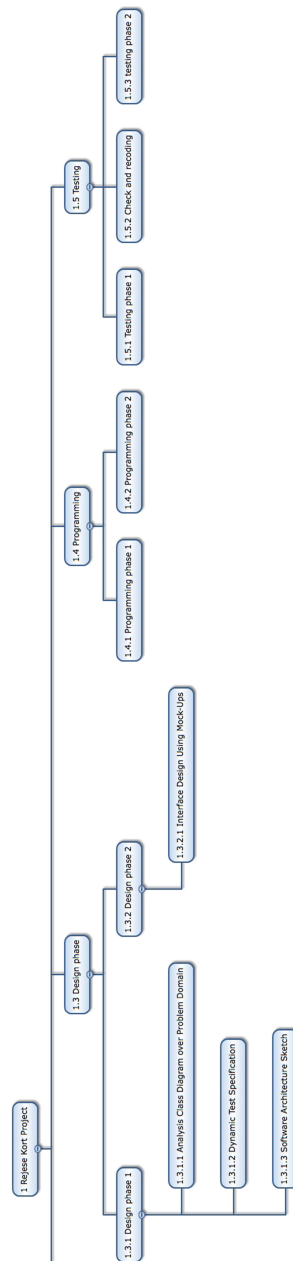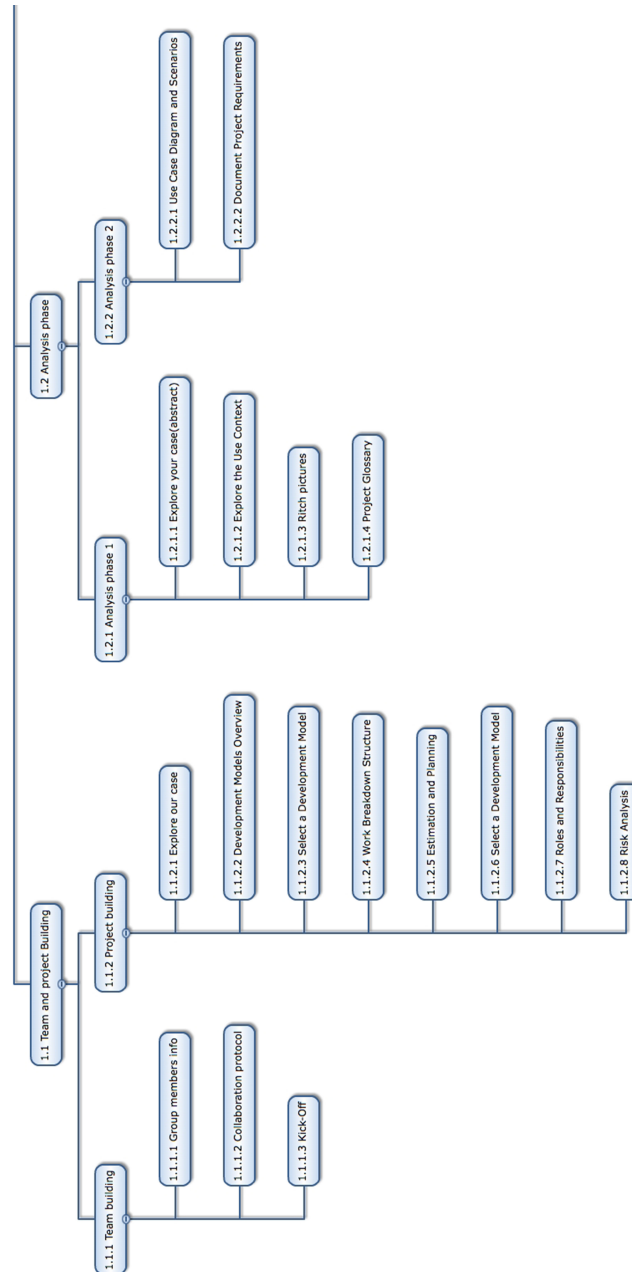
Interview Questions

There will be 3 participants

1. passagers(structured)
   a) Rejse Kort users
      i. How old are you?

         Range:15 - 25 26- 35 older

      ii. How often do you use Rejse Kort?

         A. Every day

         B. 1-3 times a week

         C. 1-3 times a month

         D. twice a year

         E. or less

      iii. Which type of Rejse kort are you using?

         A. Personal card (Rejsekort personligt)

         B. Anonymous card (Rejsekort anonymt)

         C. Flex card (Rejsekort flex)

      iv. How did you order your Rejse Kort?

         A. Online

         B. Ticket office

      v. How do you charge your Rejse Kort?

         A. Dankort agreement / automatic recharge

         B. Online

         C. Machine

         D. Ticket office

      vi. Which transportation do you use very often?

         A. Metro

         B. regular train

         C. S train

         D. bus

         E. boat

      vii. Why do you choose Rejse kort at the beginning?

         A. price

         B. new technology

         C. convenient(easy, cover all Denmark etc.)

         D. others...

      viii. What the problems are you facing when you use Rejse Kort?

         A. Checking in

         B. Checking out

         C. Loading credit/balance

         D. No, I don't have yet

         E. If yes, Please explain

         F. Other problems?

      ix. Which improvement do you expect from Rejse Kort?

         A. Open answer

   b) non user passengers

    i. How old are you?

        Range15 - 25 26- 35 older

   ii. How often do you use public transportation?

     A. Every day

     B. 1-3 times a week

     C. 1-3 times a month

     D. twice a year

     E. or less

   iii. Which type of tickets are you using?

     A. single ticket

     B. period card

     C. youth card

     D. clip card

     E. app ticket

     F. pension card

     G. other

   iv. Why are you not using Rejse Kort?

     A. Price

     B. New technology

     C. Difficult

     D. Not convenient

     E. other

   v. Which improvement would you suggest Rejser Kort then you can use it in the future?

     A. Open answer

     B. I would never use it

c) company IT professionals(semi-structured)

    i. How do you think about rejse kort system?

   ii. Which kind of problems do you face with the Rejse Kort system ?

   iii. What could be improved? and how?

   iv. How many percent of users in Denmark do your company expectful?

d) ticket controllers(semi-structured)

    i. Which kind of problems do you face working with the Rejse Kort system ?

   ii. What could be improved?

   iii. How does this card help you work? F.x make easy for your work?

   iv. Which of the transportation ticket you prefer? why?

It can be concluded from our observation and interview that the current Danish TravelCard system have issues which can be improved and corrected. Some of the problem are listed below:

⇒ Passengers did not manage to find out how to check out, or simply forgot to do so.

⇒ The fair billed was not the cheapest fare as promised in the advertisement, especially for kids and pensioners.

⇒ Passengers who were accused by ticket controllers of not paying could not proof that they had checked in though they, at least, thought they did so. Passengers do not get a receipt. With other words: errors caused by the system could not be proven to the ticket controller.

⇒ Control over billing data and privacy – also in the light of the current revelation of privacy issues with foreign and local authorities collecting data 'for stock' is an issue as well.

⇒ The sounds is coming from the machine is quite confusing to the passengers.

⇒ The machine is sometimes not functioning thus making it hard to check in or check out.

⇒ The out and check in machines look very similar.

Waterfall

Advantages

Every phase has a defined start and end point. Progress can be conclusively identified (through the use of milestones) by both vendor and client.

Disadvantages

If there are many errors in the results, developers need to go back all the processes. No user interaction. Its major problem is the inflexible partitioning of the project into distinct stages

Characteristic

The waterfall model is an example of a plan-driven process—in principle, you must plan and schedule all of the process activities before starting work on them

In principle, the waterfall model should only be used when the requirements are well understood and unlikely to change radically during system development.

Agile

Advantages

Saving time for the developing team. Customer satisfaction by rapid, continuous delivery of useful software. People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other. Working software is delivered frequently (weeks rather than months). Close, daily cooperation between business people and developers. Regular adaptation to changing circumstances. Even late changes in requirements are welcomed. Constant Learning, Knowledge Creation and Knowledge Sharing

Disadvantages

In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.

There is lack of emphasis on necessary designing and documentation.

Characteristic

Individuals and interactions over processes and tools.

Working software over comprehensive documentation.

Customer collaboration over contract negotiation.

Responding to change over following a plan.

Rational Unified Process (RUP)

Advantages

Full of the documents describing for the project. It is good for large project. There are iterations in the different phases.

Disadvantages

The team members need to be expert in their field to develop a software under this methodology.

The development process is too complex and disorganized. It is wasting time to make many documents for a small project.

Characteristic

RUP provides a full lifecycle approach covering a series of product lifecycle phases called inception, elaboration, construction, and transition.

RUP provides a software development method and a set of software engineering practices that cover the majority of software development disciplines.

RUP is incremental: Each iteration builds on the functionality of the prior iteration; the software application evolves in this fashion with the benefit of regular and continuous feedback.

Spiral

Advantages

Risk management as inbuilt feature. Changes can be done later without spending much time.

Disadvantages

High cost involved in this model. Expertise needed as product has to review and redone time to time.

Characteristic

The main difference between the spiral model and other software process models is its explicit recognition of risk. A cycle of the spiral begins by elaborating objectives such as performance and functionality.

Prototyping

Advantages

User is involved throughout the development process, which increases the likelihood of user acceptance of the final implementation.

Small-scale mock-ups of the system are developed following an iterative modification process until the prototype evolves to meet the users' requirements.

While most prototypes are developed with the expectation that they will be discarded, it is possible in some cases to evolve from prototype to working system.

Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.

Errors can be detected much earlier.

Quicker user feedback is available leading to better solutions.

Missing functionality can be identified easily

Disadvantages

Leads to implementing and then repairing way of building systems.

Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.

Incomplete application may cause application not to be used as the full system was designed.

Too much involvement of client, is not always preferred by the developer.

Too many changes can disturb the rhythm of the development team.

Characteristic

A basic understanding of the fundamental business problem is necessary to avoid solving the wrong problem.

Prototype model should be used when the desired system needs to have a lot of interaction with the end users.

Extreme Programming

Advantages

Developers write test code first. It is test driven development. The aim is using test code to design the structures of system. It is good for developers to consider the risks that could be in the structures.

It is fast to develop the software products.

Disadvantages

XP is more related to the technical people. Business users cannot be part of development.

There are few documents. It is hard to use in the large project.

XP ignore the software design.

There is no specific process for developing.

Characteristic

Extreme Programming is an Agile methodology, there are characterized to be an open for adjustments running on development. User words is law, and developers should always have users priority of, which part which must be developed next time, and approval by the fact that it developed was actually what they needed.

Incremental Development

Advantages

- Generates working software quickly and early during the software life cycle. - More flexible – less costly to change scope and requirements. - Easier to test and debug during a smaller iteration. - Customer can respond to each built. - Lowers initial delivery cost. - Easier to manage risk because risky pieces are identified and handled during it'd iteration.

Disadvantages

- Needs good planning and design. - Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.

Characteristic

In incremental model the whole requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle. Cycles are divided up into smaller, more easily managed modules. Each module passes through the requirements, design, implementation and testing phases.

Rapid Application Development

Advantages

Reduced development time.

Increases reusability of components.

Quick initial reviews occur.

Encourages customer feedback.

Disadvantages

Depends on strong team and individual performances for identifying business requirements.

Only system that can be modularized can be built using RAD.

High dependency on modeling skills.

There are times when the team ignores necessary quality parameters such as consistency, reliability and standardization.

Characteristic

RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.

It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.