POLITECNICO
MILANO 1863

Design Document - myTaxiService

Ciceri, Filippo
filippo.ciceri@mail.polimi.it

Cesaro, Federico
federico.cesaro@mail.polimi.it

Capecchi, Luca
luca.capecchi@mail.polimi.it

February 2, 2016

Contents

1 Function Points and COCOMO

1.1 Function Points

1.1.1 ILF - Internal Logic Files

The ILFs of our system will be the following:

- Guest
- User
- Taxi Driver
- Ride

1.1.2 ELF - External Logic Files

The ELFs of our system will be the following:

- External Maps

1.1.3 EI - External Input

The EIs of our system will be the following:

- Signup
- Login/Logout
- User Information Modification
- Ride Booking
- Ride Modification/Cancellation
- Driver Status Modification

1.1.4 EQ - External Enquiry

The EQs of our system will be the following:

- Ride History
- Ride Information
- User Information
- Driver Request Response

1.1.5 EO - External Outputs

The EQs of our system will be the following:

- Ride History Creation
- Shared Ride
- User Notification

1.1.6 Complexity Estimation

ILF	**Complexity**
Guest	Simple
User	Medium
Taxi Driver	Medium
Ride	High

ELF	**Complexity**
External Maps	High

EI	**Complexity**
Signup	Medium
Login	Simple
Logout	Simple
User Information Modification	Medium
Ride Booking	Medium
Ride Modification	Medium
Ride Cancellation	Simple
Driver Status Modification	Simple

EQ	**Complexity**
Ride History	Simple
Ride Information	Simple
User Information	Simple
Driver Request Response	Simple

EO	**Complexity**
Ride History Creation	High
Shared Ride	High
User Notification	Simple

This is the table we refer to assign cost to the Function Points

Function Types	**Complexity**		
	Simple	**Medium**	**High**
N.EI	3	4	6
N.EO	4	5	7
N.EQ	3	4	6
N.ILF	7	10	15
N.EIF	5	7	10

So we have:

Function Types	**Number**			**Points**
	Simple	**Medium**	**High**	
EI	4	4	0	28
EO	1	0	3	25
EQ	4	0	0	12
ILF	1	2	3	32
EIF	0	0	1	10
TOTAL				107
SLOC	$\times 53$			5671

The multiplicator to convert UFP to SLOC for Java language is 53.

1.2 COCOMO

To calculate the Effort of our project, we will use COCOMO II model. The formula that calculates the Effort is:

$$\text{Effort} = A \times EAF \times KSLOC^{E}$$

Where:

- EAF = $\prod_i C_i$ with C_i a single Cost Driver
- E = $0.91 + 0.01 \times \prod_i SF_i$ with SF_i a single Scale Driver
- KSLOC = Kilo Source Lines of Code
- A = 2.94

So now we need to analize the Scale Drivers and the Cost Drivers.

1.2.1 Scale Drivers

Scale Drivers are parameters that non-linearly influence the effort, in relation to the Lines of Code.
There are five types of these parameters, each one can go to `Very low` to `Extra high`:

- **Precedentedness:** this parameter reflects the previous experience on this type of project of the people that are working on it. `Very low` means no experience at all, `Extra high` means complete familiarity. In our project this parameter is `Low` because we have some experience in project desing but most of these issues are new to us.

- **Development flexibility:** explains the level of flexibility in the development process. `Very low` means that the developer has been given rigorous requests and clear goals, `Extra high` means that were given only generic goals. We set it to `Nominal` because we were in a not too strict situation, where we had clear goals but we could interpret them with a certain degree of freedom.

- **Risk resolution:** evaluates the risk assessment. We set it to `Nominal`, since our risk analysis is not too extensive.

- **Team Cohesion:** it obviously identifies the cohesion of the team members. In our case we set it to `High` since we know each other quite well but it's the first time we work together in a group of three people.

- **Process Maturity:** Reflects the process maturity of the organisation. The computation of this value depends on the CMM Maturity Questionnaire but we estimated it at `High` that corresponds to CMM Level 3.

Code	**Name**	**Factor**	**Value**
PREC	Precedentedness	Low	4.96
FLEX	Development flexibility	Nominal	3.04
RESL	Risk resolution	Nominal	4.24
TEAM	Team cohesion	High	2.19
PMAT	Process maturity	High	3.12
Total	E $= 0.91 + 0.01 \times \prod_i SF_i$		1.09

1.2.2 Cost Drivers

COCOMO II has 17 Cost Drivers that are multiplicative factors that reflect some characteristics of the developing process. The range of the values is the same as the Scale Drivers. The table below shows our project's values for the Cost Drivers.

Code	**Name**	**Factor**	**Value**
RELY	Required Software Reliability	High	1.10
DATA	Data base size	Nominal	1.00
CPLX	Product Complexity	Nominal	1.00
RUSE	Required Reusability	High	1.07
DOCU	Documentation match to life-cycle needs	Low	0.91
TIME	Execution Time Constraint	Very High	1.29
STOR	Main Storage Constraint	Nominal	1.00
PVOL	Platform Volatility	Low	0.87
ACAP	Analyst Capability	Nominal	1.00
PCAP	Programmer Capability	Nominal	1.00
APEX	Application Experience	Very Low	1.22
PLEX	Platform Experience	Nominal	1.00
LTEX	Language and Tool Experience	Low	1.09
PCON	Personnel Continuity	High	0.90
TOOL	Usage of Software Tools	Nominal	1.00
SITE	Multisite Development	High	0.93
SCED	Required Development Schedule	High	1.00
Total	$\mathrm{EAF} = \prod_i C_i$		1.34

1.2.3 Effort Estimate

Our effort estimate amounts to **26.1 person-months**.

1.2.4 Duration

The Duration of our project is given by this equation:

$$\text{Duration} = 3.67 \times (\text{Effort})^{0.28+0.2\times(E-0.91)}$$

Where the variables are the same as in the Effort Equation.

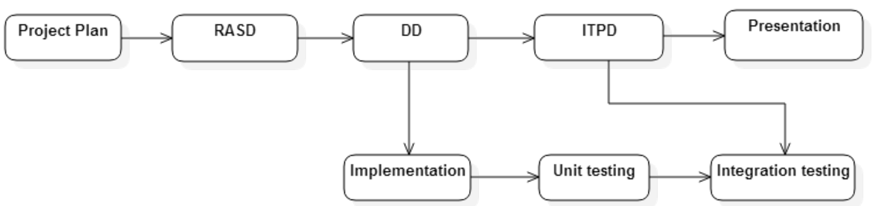
The estimate of the duration of the project amounts to **10.2 months**. This estimate gives us a team size of 2.56 people. With these numbers in mind, we can expect a 10 months time frame to be a generous estimate, due to our team size being 3.

2 Task Schedule

The tasks that we have to complete for this project can be divided in:

1. Deliver the main documents concerning the project such as:
    - The *Requirement Analysis and Specification Document* (RASD) containing the domain assumptions, the goals and the requirements of our project.
    - The *Design Document* (DD) that contains the explanation of the architecture of our system.
    - The *Integration Testing Plan Document* containing the plan of integration testing on our system.
    - This *Project Plan*
2. A short presentation about the previous documents
3. Implement the following parts of the service:
    - The application front-end for both mobile and web platforms.
    - The ride management system.
    - The ride reservation system.
    - The user management system.
4. Unit and Integration testing, which can be done iteratively as each part of the service is developed, and can be actively used as development feedback before release.

The tasks to be performed do not require a strict order, especially with regards to the development of the software. However, we did generate a dependecy graph to show what parts have to be completed before work can be started on other parts. The dependency graph is pictured below.



Due to the nature of our project, there are several fixed deadlines with regards to the delivery of most documents. However, the development and testing cycle has no set deadlines yet, since it is too early in the development cycle to assess the amount of time necessary to develop each part of the software.

Activity	**Start Date**	**Deadline**
Project Plan	29/09/2015	14/10/2015
RASD	15/10/2015	06/11/2015
DD	07/11/2015	04/12/2015
ITPD	04/01/2016	21/01/2016
Implementation and Testing	23/01/2016	29/07/2016
Presentation	10/02/2015	15/02/2015

Table 1: Schedule for project tasks.

3 Resource Allocation

The resource allocation we have done is tentative and in no way definitive. Due to the needs of the team, we may dynamically reallocate people to different parts of the project. Currently, we are only putting high level tasks, since being fiscal with micro-tasks does not make sense in a large project.

In this current iteration, it is important to note we have assigned every member to every macro-task, so each person in our small team has a proper grasp on the entire project, as opposed to specializing in a specific part of the service. This ensures better options for maintenance and updates to the service.

Project Plan		
Person	29/09/2015 to 14/10/2015	
	Week 1	**Week 2**
Filippo	FP and Tasks	COCOMO
Federico	FP and COCOMO	Resource Allocation
Luca	Risk	Risk

RASD				
Person	15/10/2015 to 06/11/2015			
	Week 1	**Week 2**	**Week 3**	**Week 4**
Filippo	Introduction	Overall Description	Scenarios	Revision
Federico	UML	Alloy	Alloy	Appendix and Revision
Luca	Mockups	Requirements	Requirements	Revision

DD		
Person	07/11/2015 to 04/12/2015	
	Week 1	**Week 2**
Filippo	Algorithm Design	Runtime View and High Level Components
Federico	UX	Component View
Luca	Sequence Diagrams	Sequence Diagrams and Runtime View
	Week 3	**Week 4**
Filippo	Component Interfaces	Algorithm Diagrams
Federico	Requirements Traceability	Revision
Luca	System Architecture	Revision

ITPD			
Person	04/01/2016 to 21/01/2016		
	Week 1	**Week 2**	**Week 3**
Filippo	Test Steps	Testing Tools	Appendix
Federico	Testing Strategy	Tests	Review
Luca	Scope	Purpose	Test Data Requirements

Implementation and Testing			
Person	23/01/2016 to 29/07/2016		
	1st Month	**2nd Month**	**3rd Month**
Filippo	Business Tier	Business Tier	Business Tier Testing
Federico	Mobile Client	Web Client and Client Testing	Intra-Component Communication
Luca	Database	Database testing	Credential and User Management
	4th Month	**5th Month**	**6th Month**
Filippo	Client Testing	Integration Testing	Final Fixes and Revision
Federico	Business Tier Testing	Deployment Test	Final Fixes and Revision
Luca	User Testing	Integration Testing	Final Fixes and Revision

Presentation	
Person	10/02/2015 to 15/02/2015
	Week 1
Filippo	Slides
Federico	Slides
Luca	Slides

4 Risks associated with the project

4.1 Project Risk

Here is a brief description of risks we may run into due to poor design.

Delays over expected deadlines: the project may be harder than we estimated. This may cause a delayed release or a release of a incomplete version.

Lack of experience: the team may lack in programming experience this may cause a possible release delay.

Structural changes: during development we may find our project guide line hard to be implemented and we may have to restructure.

Lack of communication: to develop this project in time the team needs to communicate; this can be hard because team members may work remotely. This can lead to misunderstandings. To minimize this we have to make a RASD and DD to guide the development.

Risk	**Probability**	**Effects**
Delays Over Expected Deadlines	High	Moderate
Lack of Experience	High	Moderate
Structural Changes	Low	Moderate
Lack of Communication	High	Moderate

Table 2: Evaluation of Project risks.

4.2 Technical Risk

Here is a brief description of risks we may have due to poor implementation.

Server downtime: we may underestimate the system load or find software bugs. This can cause server downtime. We can work around the software bugs with load testing or by using third party cloud-based servers with redundancy.

Market reach: the end user may not want to use our taxi service.

Data security: we can have data lost or leaked by hardware failure, software bugs or third party attacks. We can avoid this by using security standards for our software and by testing it.

Scalability: the system could have problem with a larger number of users. We can use a third party cloud server to host our system and aid the growth process.

Bad code: the code in large project may become hard to read/understand. We can avoid this by write a Design Document to guide our development.

Risk	**Probability**	**Effects**
Server Downtime	Moderate	High
Market Reach	Moderate	Moderate
Data Security	Moderate	High
Scalability	Low	Moderate
Bad Code	Moderate	Moderate

Table 3: Evaluation of technical risks.

4.3 Economical Risk

Bankruptcy the city may withdraw their offer. We can't avoid this.

Wrong cost evaluation the cost may exceed our budget. To prevent this we may run a very in-depth analysis to prevent this risk.

Risk	**Probability**	**Effects**
Bankruptcy	Low	High
Wrong Cost Evaluation	Moderate	High

Table 4: Evaluation of economical risks.

5 Appendix

5.1 Software and Tools Used

- MiKTeX (http://www.miktex.org/) to format and create this document.
- StarUML (http://www.staruml.io/) used to create the dependency chart.

5.2 Work Hours

Time spent on the creation of this DD:

- Filippo Ciceri: 5 hours
- Federico Cesaro: 7 hours
- Luca Capecchi: 4 hours