POLITECNICO DI MILANO
Computer Science and Engineering
Software Engineering 2 Project

Tesla Car Sharing

Project Plan Document

Danchenko Elena, 874840
Cilloni Stefano, 880924

January 22nd, 2017

Document version: 1.0

Contents

1. Function points estimation **3**
1.1 Brief introduction to functional points and complexity levels 3
1.2. Internal Logic Files 4
1.3. External Interface Files 5
1.4. External Inputs 5
1.5. External Outputs 7
1.6. External Inquiries 8
1.7 Final estimation 9
1.8. UFP to SLOC conversion 9

2. COCOMO II estimation **10**
2.1. Scale Drivers 10
2.2. Cost Drivers 11
2.3. Effort estimation 12
2.4. Duration 12

3. Tasks and schedule **13**

4. Resource allocation to tasks **15**

5. Risks associated to the project **18**
5.1. Project risks 18
5.2. Technical risks 19
5.3. Economical risks 20

6. References **21**

7. Hours of work **22**
7.1. Elena Denchenko 22
7.2. Stefano Cilloni 22

8. Changelog **23**

1. Function points estimation

1.1 Brief introduction to functional points and complexity levels

We report two tables of COCOMO II: the first one expose how to evaluate complexity of Function Points based on the number of data elements and the second one shows the UFP complexity weights.

Table 2. FP Counting Weights

For Internal Logical Files and External Interface Files

	Data Elements		
Record Elements	**1 - 19**	**20 - 50**	**51+**
1	Low	Low	Avg.
2 - 5	Low	Avg.	High
6+	Avg.	High	High

For External Output and External Inquiry

	Data Elements		
File Types	**1 - 5**	**6 - 19**	**20+**
0 or 1	Low	Low	Avg.
2 - 3	Low	Avg.	High
4+	Avg.	High	High

For External Input

	Data Elements		
File Types	**1 - 4**	**5 - 15**	**16+**
0 or 1	Low	Low	Avg.
2 - 3	Low	Avg.	High
3+	Avg.	High	High

Figure 1: Table2. FP counting weights.

Table 3. UFP Complexity Weights

	Complexity-Weight		
Function Type	**Low**	**Average**	**High**
Internal Logical Files	7	10	15
External Interfaces Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

Figure 2: Table3. UFP complexity weights.

1.2. Internal Logic Files

The system includes some ILFs that, defined as homogeneous sets of data, are used to store information with persistence. The ILFs of our system correspond to entities created and modified in the Datastore. Therefore the system includes the following ILFs:

- **Software user.** This entity has the fields: *key*, user_type, name, surname, password_salt, password_hash, email, phone_number, birthdate, CF, country_of_birth, city_of_residence, street, house_nuber, CAP, identity_document_type, identity_document_number, driver_license_number, driver_license_expiration_date, pc_brand, pc_number, pc_expiration_date, pc_CCV2.
  Some of these fields are left empty for the software users that are not clients. They are: Administrator, managers and operators.
- **Car.** This entity has the fields: *key,* model, license_plate_number, status
- **Task.** This entity has the fields: *key,* car_key, operator_key, title, type, description, subtask_list, creation_date, acceptance_date
- **Ride.** *key*, car_key, user_key, receipt_key, begin_time, end_time, number_of_passenger, start_location, end_location, destination
- **Receipt.** *key*, car_key, user_key, applied_disciunts_list, fees_list, overchanges_list, total_amount, payment_method, date

ILF	**Complexity**	**FP**
Software user	Avg.	10
Car	Low	7
Task	Low	7
Ride	Low	7
Receipt	Avg.	10

Table 1: Functional points of ILFs

1.3. External Interface Files

The system manages data also coming from external applications in order to achieve its goals. These information are categorized as External Interface Files (EIFs). We list below those external application, and the data handled by the system:

- **Google Maps API**. Information necessary to show maps to the users and to calculate distances required from some functions of the system.
- **Car Control Software API**. Data concerning the car status and user activities near and inside the car.
- **Banking System API**. Payments information generated by the bank system and sent to our software to communicate payment outcomes.

EIF	**Complexity**	**FP**
Google Maps API	Avg.	7
Car Control Software API	Avg.	7
Banking System API	Avg.	7

Table 2: Functional points of EIFs

1.4. External Inputs

An External Input (EI) is defined as an elementary operation done by the user to manage Internal Logic Files through the ability to add, change and delete the data.
The EIs of the system are:

- User registration
- User login
- Create an administrative (managers, operators) account
- Modify an administrative account
- Remove an administrative account
- Create a car record
- Modify a car record
- Remove a car record
- Task acceptance
- Task waiving
- Task completion
- Modify the cost of the ride

- Modify the fee applied for the expired car reservation
- Send car reservation
- Cancel car reservation
- Submit a payment for a receipt

EIs	**Complexity**	**FP**
User registration	Avg.	4
User login	Low	3
Create an administrative account	Low	3
Modify an administrative account	Low	3
Remove an administrative account	Low	3
Create a car record	Low	3
Modify a car record	Low	3
Remove a car record	Low	3
Task creation	Low	3
Modify the cost of the ride	Low	3
Modify the fee for the expired car reservation	Low	3
Send a car reservation	Low	3
Cancel car reservation	Low	3
Submit a payment for a receipt	Avg.	4

Table 3: Functional points of EIs.

1.5. External Outputs

An External Output (EO) is defined as an elementary operation that allow the user to produce outputs. It usually includes the elaboration of data from logic files.
The EOs of the system are the following:

- Produce a the list of searched car
- Produce a mail notification due to some other actions
- Produce a mobile notification
- Produce a command for the car control software

EOs	**Complexity**	**FP**
Search for a car	Avg.	5
Mail notification	Low	4
Mobile notification	Low	4
Command for the car control software	Low	4

Table 4: Functional points of EOs.

1.6. External Inquiries

Thanks to a computerized system a user can at the end display specific data from files with ease. To accomplish this, a user inputs selection information that is used to retrieve data that meets the specific criteria. In this situation there is no manipulation of the data. It is a direct retrieval or raw calculation on/of information contained on the files. These transactions are referred to as External Inquiries (EQ).
The system has following EQs:

- Search for a car
- Show the maps
- Indirect sending of commands to car control software
- Task acceptance
- Task completion
- Task waiving

EQs	**Complexity**	**FP**
Search for a car	Avg.	4
Show the maps	Low	3
Indirect sending of commands to car control software	Low	3
Task acceptance	Low	3
Task completion	Low	3
Task waiving	Low	3

Table 5: Functional points of EQs.

1.7 Final estimation

With the following table we summarize the estimated Function Points assigned in the paragraphs before.

Function type	**FP**
Internal Logic Files	41
External Interfaces Files	21
External Input	44
External Output	17
External Inquiries	19
TOTAL	142

Table 6: Summary of Functional Points.

1.8. UFP to SLOC conversion

In COCOMO II Model Definition Manual [1] there is no multiplicator factor to convert the Unadjusted Function Points to Source Lines of Code for Python. That's why because Python combines features of 4th and 3rd generation languages we decided to evaluate it as 35.

So, depending on the conversion rate, we have a

$$SLOC = 142 \bullet 35 = 4970$$

2. COCOMO II estimation

2.1. Scale Drivers

Scale drivers are some of the most important factors contributing to a project's duration and cost estimations that also reflect the non-linearity of the effort with relation to the SLOC. They are a significant source of exponential variation on a project's effort and productivity variation.

Code	Name	Factor	Value
PREC	Precedentedness	Nominal	3.72
FLEX	Development flexibility	Nominal	2.03
RESL	Risk resolution	High	4.24
TEAM	Team cohesion	High	2.19
PMAT	Process maturity	High	3.12
Total	$E = 0.91 + 0.01 \times \Sigma i\ (\ SFi\ )$		1.063

Table 7: Scale Drivers for the project.

Precedentedness.

It reflects the previous experience of the organization with this type of project. Very low means no previous experience, Extra high means that the organization is completely familiar with this application domain. The precedentedness for our team is set to an average level because we have already developed some medium sized projects that use technologies used in this project. Also we have some knowledge about software development in general, so this is the reason because we evaluated precedentedness driver as Nominal.

Development Flexibility.

High if there are no specific constraints to conform to pre-established requirements and external interface specs. In our case there is prescribed process we have to follow but we have some degree of freedom to define the design and development of our system. We set this driver to Nominal.

Risk Resolution.

High if we have a good risk management plan, clear definition of budget and schedule, focus on architectural definition. We conducted analysis on risks and their impact mitigation further in this document, so we evaluate this driver as high.

Team Cohesion.

It reflects how well the development team know each other and work together. We set this driver to High since the cohesion among us is pretty good.

Process Maturity.

Refers to a well known method for assessing the maturity of a software organization, CMM, now evolved into CMMI (see additional material).

We evaluated our project as being Level 3, so the we set the driver as high.

2.2. Cost Drivers

The initial estimates made in the COCOMO II model are adjusted using a set of attributes (project cost drivers) that reflect:

1. Product characteristics such as the required system reliability and product complexity.
2. Computer characteristics such as execution time or memory constraints. These are constraints imposed on the software by the hardware platform.
3. Personnel characteristics such as programming language skills that take the experience and capabilities of the people working on the project into account.
4. Project characteristics of the software development project such as the IDE that is available and the development schedule.

The cost drivers for the system are the following:

Code	**Name**	**Factor**	**Value**
RELY	Required Software Reliability	Nominal	1.00
DATA	Data base size	Nominal	1.00
CPLX	Product Complexity	Nominal	1.00
RUSE	Required Reusability	Nominal	1.00
DOCU	Documentation match to life-cycle needs	Nominal	1.00
TIME	Execution Time Constraint	Nominal	1.00
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	Low	1.09
LTEX	Language and Tool Experience	Nominal	1.00
PCON	Personnel Continuity	Very high	0.81
TOOL	Usage of Software Tools	Nominal	1.00
SITE	Multisite Development	Very High	0.93
SCED	Required Development Schedule	Nominal	1.00
Total	EAF = ∏i (Ci)		0.872

Table 8: Cost Drivers for the project.

2.3. Effort estimation

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

Where:

- A = 2.94
- EAF = ∏i (Ci) Product of al cost drivers
- KSLOC Kilo Source Line of Code estimated in the 1.8 UFP to SLOC conversion
- E = 0.91 + 0.01 × Σi (SFi) Factor depending on the scale drivers

The total calculated effort results in **14.0958301 person-month.**

2.4. Duration

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

Where:

- PM Is the effort calculated in the paragraph 2.3 Effort estimation.
- E Is the the exponent depending on the scale drivers (the same one of the effort equation).

The duration obtained from the above formula results in a total of **8.34 months**.

Our team has two members so the duration of the project is supposed to be

$$\frac{14.0958}{2} = 7.048$$

In order to be careful with the task scheduling, we will approximate it to 7.5 months.

3. Tasks and schedule

The main tasks which constitute this project include:

1. Delivery of Requirement Analysis and Specification Document
2. Delivery of Design Document
3. Delivery of Integration Testing Plan Document
4. Delivery of Project Management Plan
5. Preparation of presentation regarding this project
6. Implementation of the software system
7. Integration testing of the system

The schedule of the first five tasks is defined by the assignment document regarding project rules and assigned date of presentation [2].
The implementation and integration tasks are scheduled according to COCOMO estimation. The integration tasks will start in parallel at the ending of the implementation in order to be sure to make great test and to end the implementation resolving any possible error.
The schedule of the project is presented in the following table.

Activity	Start Date	Deadline
RASD	2016-10-16	2016-11-13
DD	2016-11-15	2016-12-11
ITPD	2017-01-03	2017-01-15
PPD	2017-01-12	2017-01-22
Presentation	2017-01-30	2017-02-09
Implementation	2017-02-10	2017-05-10
Integration Testing	2017-04-25	2017-05-25

Table 9: Schedule for project tasks.

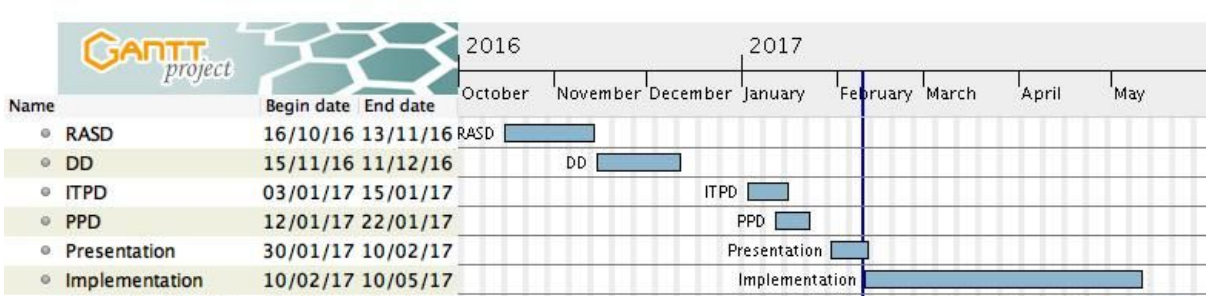

Figure 1: Gantt chart of the project.

4. Resource allocation to tasks

The following tables want to show how the available resources are allocated to implement the project.
The tasks are reported by grouping them with an high level of abstraction because managing too finely the work of people on a large project is of little use.
With the resulting schedule both the team members have a general overview on the whole project: each task involves both the members. This kind of schedule causes the increase of the time needed to complete the project, but it also induces both the member to be aware of the work of the other in order to have at least one different point of view on each problem.

	Elena	**Stefano**

RASD (from 2016-10-17 to 2016-11-13)

Week 1	Overall description	Requirements
Week 2	Use cases, System features	Overall description
Week 3	Modelling	Scenario
Week 4	Revision of Stefano work	Revision of Elena work

DD (from 2016-11-15 to 2016-12-11)

Week 1	Overall description	Architectural design
Week 2	Architectural design, Algorithm design	Architectural design, Algorithm design
Week 3	Requirement traceability, Revision of Stefano work	User interface design, Revision of Elena work

ITPD (from 2017-01-03 to 2017-01-15)

Week 1	Integration strategy, Individ Test description	Test description, Individual Step and Test description
Week 2	Program stubs and driver, Revision of Stefano work	Program stubs and driver, Revision of Elena work

	Elena	Stefano

PPD (from 2016-01-12 to 2017-01-22)

Week 1	Function points estimation, Risks associated to the project	Function points estimation, COCOMO II estimations
Week 2	Task and schedule, Revision of Stefano work	Resource allocation to task, Revision of Elena work

Presentation (from 2017-01-30 to 2017-02-09)

Week 1	Slides	Slides
Week 2	Presentation	Presentation

Implementation (from 2017-02-10 to 2017-05-10)

Week 1	Initial setup of the development environment	Initial setup of the development environment
Week 2	authentication module development and configuration	authentication module development and configuration
Week 3	Implementation of car functional module	Implementation of task functional module
Week 4	Implementation of car functional module	Implementation of car functional module
Week 5	Implementation of the ride functional module	Implementation of notification functional module
Week 6	Implementation of payment functional module	Implementation of the ride functional module
Week 7	API development for the car and payment functional modules	API development for the ride and task functional modules
Week 8	Integration with Bank Software	API development for the car and payment functional modules
Week 9	Integration with Bank Software	Integration with Car Control Software
Week 10	Mobile application development	Mobile application development
Week 11	Mobile application development, setup of testing environment	Mobile application development, setup of testing environment

Week 12	Mobile application development, code corrections to fix errors raised by tests	Mobile application development, code corrections to fix errors raised by tests
Integration testing (from 2017-04-25 to 2017-05-25)		
Week 1	Initial setup of the testing environment, unit tests	Initial setup of the testing environment, unit tests
Week 2	Development of drivers and stubs required	Development of drivers and stubs required
Week 3	Integration testing	Integration testing
Week 4	Integration testing	Integration testing

Table 10: Resource allocation.

5. Risks associated to the project

The project is associated with many project, technical and economical risk

5.1. Project risks

Personnel shortfalls.

Key staff are ill at critical times in the project. To minimize the effect of that risk factor the personnel knows the tasks of each other and so each employee is more or less substitutable.

Unrealistic time and cost estimations.

In order to reduce the impact of mistakes in time and costs estimations the values of the primal estimation were a little bit heightened. In case of serious miscalculation of that estimates, the solution to solve the problem might be to try to start a negotiation with the project ordering company in order to correct the teatry.

Late changes to requirements.

To be resilient to changes to requirements the system is being designed as easily extendable. Functional modules are the key points of the extendability and modularity that allow us to be resilient in changes on what parts of the software have to do.

Risk	**Probability**	**Effects**
Personnel shortfalls	Moderate	High
Unrealistic time and cost estimations	Moderate	Catastrophic
Late changes to requirements	Very Low	Low

Table 11: Evaluation of project risks.

5.2. Technical risks

Integration testing failure.

In our project this risk is minimized by using incremental approach for integration and precise integration plan. Also for minimization of this risk the interfaces between components should be precisely described.

Downtime.

Since the platform chose on which develop the software is a cloud platform there is a very low probability that the system can go offline for load or for server failures. The distributed architecture of the *Paas* on which the software relay is fault tolerant by definition and can automatically manage load balancing for extra load.

Poorly structured code.

With the growth of the project, the code may become overloaded, badly structured and unreadable. This risk is minimized by writing a good Design Document before the start of implementation and by the adoption of the functional modules structure.

Data loss.

Most of database management systems suffer of this problem. We chose for the implementation of the data tier to use the integrated database management system provided by the Google Cloud Platform. The Datastore is fault tolerant and this feature decrease a lot the possibility of data loss. For prevention we have plan to schedule some backups through triggers offered by the platform.

Data leaks.

Bad software configurations or leak of security standard adoption can be the causes of data theft and this may expose personal user data to the world. This risk must be prevented by adopting industry security standards, by encrypting communications and by doing regular penetration testing on the system.

Risk	**Probability**	**Effects**
Integration testing failure	Low	High
Downtime	Very Low	High
Poorly structured code	Moderate	Moderate
Data loss	Very Low	Catastrophic
Data leaks	Moderate	Catastrophic

Table 12: Evaluation of technical risks.

5.3. Economical risks

Delay of payment from our project's ordering company.

We can try to deal with temporary financial difficulties (of course if we are quite sure that delay is just temporary) by rearranging financial flows from other projects of our company.

Bankruptcy.

The income from the sales of the software may not be enough to sustain the development, maintenance and deployment of the software system. A good feasibility study helps to avoid this situation.

Competitors.

It could happen that other companies may develop the same project in parallel to us and then supply the same product at a lower price. This will put out of the market our software. The software must be thought to be competitive and innovative functions should be introduced during its life.

Risk	**Probability**	**Effects**
Delay of payment	Low	High
Bankruptcy	Very Low	High
Competitors	Moderate	High

Table 12: Evaluation of economical risks.

6. References

[1] COCOMO II Model Definition Manual - http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

[2] Specification document: Assignments AA 2016-2017.pdf

7. Hours of work

7.1. Elena Denchenko

- 12/01/2017: 2 h
- 14/01/2017: 1 h
- 15/01/2017: 2 h
- 17/01/2017: 2 h 30 min
- 19/01/2017: 1 h 30 min
- 22/01/2017: 1 h

7.2. Stefano Cilloni

- 12/01/2017: 2 h
- 14/01/2017: 30 min
- 15/01/2017: 1 h 30 min
- 17/01/2017: 1 h
- 20/01/2017: 2 h
- 22/01/2017: 2 h

8. Changelog