Politecnico di Milano

A.A. 2015-2016

Software Engineering 2 Project

myTaxiService

Project Plan Document

Ver. 1

February 2, 2016

Alessio Martorana - 860584

1 Project Size and Effort Estimation

In this chapter, the project size and effort estimation is performed following "function points" and "COCOMO" approaches.

1.1 Function points

Function points are a way to measure and express the size of the business functionality of an information system. Based from an evaluation of program characteristics and functionalities, function points technique gives a numeric estimation of size of the business functionality.

In Function points estimation technique, information system's functions are divided into the following sets, which are called the "function types":

- Internal Logical File (ILF): homogeneous set of data used and managed by the application
- External Interface File (EIF): homogeneous set of data used by the application but generated and maintained by other applications
- External Input: Elementary operation to elaborate data coming from the external environment
- External Output: Elementary operation that generates data for the external environment
- External Inquiry: Elementary operation that involves input and output

Some amount of points are associated to each set in the list based on the complexity of the system's function type like in the following table. Complexity value has to be selected among: "Simple", "medium" and "complex" and, of course, is evaluated on the characteristics of the application.

Function Types	**Weight**		
	Simple	Medium	Complex
N. Inputs	3	4	6
N. Outputs	4	5	7
N. Inquiry	3	4	6
N. ILF	7	10	15
N. EIF	5	7	10

Every function of the system has to be associated to one of these sets based on its complexity and then, taking the sum of all the amounts of points, the Unadjusted Function Points (UFP) are obtained.

1.2 Function Points calculation for myTaxiService system

For each function type presented in the previous table, in this paragraph the categorization of the functions of the system in the function types and function points calculation shown.

1.2.1 ILFs

ILF	Complexity	Justification	Function Points
User accounts handling	Simple	The only need is to handle user data, that is handling a simple data structure: Name, surname, email and password of the user	7
Taxi Drivers accounts	Simple	Again it is only needed to handle a very simple data structure (actually only taxi code and password fields)	7
Requests handling	Simple	A request has a simple structure (a few fields) and no complex operations are going to be done on it	7
City Zones handling	Complex	Based on the dimension and or shape of the city and on the optimization developers might want to achieve in their solution, city zone handling could become complex	15
Taxi queues handling	Medium	Taxi driver queues handling algorithms could become a little bit complex	10
Total Function Points			**46**

1.2.2 EIFs

ILF	Complexity	Justification	Function Points
Geografic map data handling	Medium	Some operations over the data received from the external service could have to be done	5
Total Function Points			5

1.2.3 External Inputs

External Input	Complexity	Justification	Function Points
Users login	Simple	Login function is a simple operation, no hard operations are requested	3
Taxi Drivers login	Simple	“	3
Users logout	Simple	Logout function is a simple operation, no hard operations are requested	3
Taxi driver logout	Simple	“	3
Users registration	Simple	Either registration is a simple operation, especially with few fields	3
User ride request	Medium	This function involves many operations such. input validation, city zone retrieving etc…	4
Taxi driver availability update	Medium	It involves city zone detecting too	4
User’s ride request acceptance/refuse	Simple	It involves the communication of the information to the taxi driver too, but should be a simple, short operation	3
Taxi driver ride request acceptance/refuse	Simple	This time the information should be delivered to the user too. But again: a simple and short operation	3
Total Function Points			**29**

1.2.4 External Outputs

External Output	Complexity	Justification	Function Points
Sending data about requested ride to the user	Simple	Not a hard function, it only needs some data forwarding	4
Sending data about requested ride to the taxi driver	Medium	It is needed to find the taxi driver to who the ride request will be forwarded	5
Total Function Points			**9**

1.2.5 External Inquiries

External Inquiries	Complexity	Justification	Function Points
Total Function Points			**0**

1.2.6. Total Function points

Adding all the previously total function points associated to the function types, we can obtain the total Function Points for the whole system:

Function type	Total Function Point
ILF	46
EIF	5
External Inputs	29
External Outputs	9
External Inquiry	0
Total Function Points	**89**

1.2 COCOMO

COnstructive COst MOdel (COCOMO) is an approach to estimate effort in a software system. COCOMO approach is based on an algorithmic non-linear model which takes into account characteristics of product, people and process in order to estimate the correspondent effort needed.

Due to the fact that COCOMO model was found too optimistic, the COCOMO II model was realeased.

COCOMO II is based on the following main parameters:

- **Source Line of Code (SLOC)**
- **Scale Drivers**
- **Cost Drivers**
- **Effort Equation**
- **Effort Adjustment Factor**
- **Schedule Equation**

1.2.1 *Source Line of Code (SLOC)*

COCOMO bases its calculations on estimates of project's size expressed as Source Line of Code (SLOC). SLOC definiment rules are:

- Only source lines delivered as part of the product are included (e.g. no test drivers and support software)
- Only sources lines created by the project staff are included (no source code generated by application generators)
- One SLOC represents a logical line of code
- Declaration are counted as SLOC
- Comments are not counted as SLOC

1.2.2 *Scale Drivers*

Scale Drivers are very important factors which contribute at project's duration and cost in COCOMO II calculations, in fact, scale drivers are used to determine the exponent which is used in the Effort Equation (see further in the document)

In COCOMO II 5 Scale Drivers are defined:

- Precedentedness
- Development Flexibility
- Architecture / Risk Resolution
- Team Cohesion
- Process Maturity

Scale Drivers are very important parameters which contribute at project's duration and cost in COCOMO II calculations, in fact, scale drivers are used to determine the exponent which is used in the Effort Equation (see further in the document).

Precedentedness

Represents the previous experience of the organisation with the type of the project. Very models no previous experience, extra high means that the organisation is absolutely familiar with this application domain.

Development Flexibility

Models the degree of flexibility in the development process. Very low means a prescribed and prefixed process is used; extra high means that the client only sets general goals and the development process is highly subject to changes.

Architecture / Risk Resolution

Reflects the width of risk analysis performed. Very low means a small grade of risk analysis, extra high means a complete risk analysis.

Team Cohesion

Represents goodness of development team work. Very low means very difficult interactions, E\xtra high indicates a team who works very well together.

Process Maturity

Represents process maturity of the organization. The computation of this value depends on the Capability Maturity Mode (CMM) Maturity Questionnaire, but an estimate can be achieved by extracting the CMM process maturity level from the CMM model 5 levels list.

1.2.3 *Cost Drivers*

Cost Drivers are multiplicative factors which determine the effort required to complete the software project.
COCOMO associates at each Cost Driver an Effort Multiplier associated with each Cost Driver rating.

1.2.4 *Effort Equation*

COCOMO II estimates of required effort (expressed in Person-Month – PM) through the following Effort equation:

$$Effort = 2{,}94 \cdot EAF \cdot (KSLOC)^E$$

Where:

- *EAF* is the Effort Adjustment Factor, which is derived from Cost Drivers.
  It is calculated as the product of the effort multipliers corresponding to each of the cost drivers for the project:

  $$EAF = \prod_{j=1}^{17} EM_j$$

  With EMj the effort multipliers corresponding to each of the cost drivers for the project.

- *E* is the exponent derived from the Scale Drivers

- *KSLOC* are the estimated Source Lines Of Codes (1 KSLOC = 1000 SLOC)

1.2.5 *Schedule Equation*

This equation is used by COCOMO to estimate the number of months required to complete a software project. The duration of a project is predicted through the following equation, on the base of effort predicted by the effort equation.

$$Duration = 3{,}67 \cdot (Effort)^{SE}$$

Where:

- *Effort* is the Effort obtained by the Effort equation
- *SE* ss the schedule equation exponent derived from the Scale Drivers

Whole COCOMO II documentation can be found in COCOMO II manual:
http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

1.3 COCOMO II Analysis for myTaxiService system

For the COCOMO II analysis of myTaxiService system I used the COCOMO II online calculator on the “Center for Systems and Software Engineering” (CSSE) website: http://csse.usc.edu/tools/COCOMOII.php.
The calculator supports both Function Points and SLOC input for describing project size. Since I introduced COCOMO approach starting from a SLOC based project size measurement, I decided to use this type of input.

1.3.1 SLOC Estimation

The SLOC needed for the project are estimated on the base of the Function Points and the programming language which is planned to be used.
In order to estimate the number of SLOC for my project through a table (described here: http://www.qsm.com/resources/function-point-languages-table) which associates a multiplicative factor to each programming language so that multiplying that factor for the total number of Function Points, the average SLOC can be obtained.
In my case I planned to develop my project using the JEE platform, so I obtained a multiplicative factor of 46.
So my estimated SLOC are:

$$SLOC = 46 \cdot 89 = 4094$$

1.3.2 Scale Drivers

Scale drivers value for MyTaxiService, together with their justification, are stated in the table below

Scale Driver	Value	Justification
Precedentedness	Very Low	I’ve never got involved in a taxi management system before
Development Flexibility	High	Since the first assignment of the project was very open to assumption and interpretations, I expect a very high rate of flexibility
Architecture / Risk Resolution	Low	In this college project is difficult to make a very thorough risk analysis, e.g. due to its intrinsic lack of links with commercial and business layers
Team Cohesion	Extra High	Since the project has only one person involved, this value must be the highest
Process Maturity	Nominal	I think that the development process of MyTaxiService system is at the 3rd level of the CMM 5 levels. This is the level at which “there are sets of defined and documented standard processes established and subject to some degree of improvement over time”

1.3.3 Cost Drivers

Scale drivers value for MyTaxiService, together with their justification, are stated in the table below

Cost Driver	Value	Justification
Required Software Reliability	Nominal	An average reliability requested
Data Base Size	Nominal	An average DB size requested
Product Complexity	Nominal	An average complexity estimated
Required Reusability	High	Very high probability of reuse of the system
Documentation match to life-cycle needs	High	Project very well documented
Execution Time Constraint	Nominal	No particular time constraint needed
Main Storage Constraint	High	Use of the main storage of the system could be very high
Platform Volatility	Nominal	The set of hardware and soft-ware elements the software uses to perform its tasks has an average tendency to change
Analyst Capability	Nominal	Average skilled analyst
Programmer Capability	Nominal	Average skilled programmer
Applications Experience	Nominal	Average experience of the team member on distributed applications
Platform Experience	Low	Low understanding of the platform
Language and Tool Experience	Nominal	All in all average level of programming language and software tool used for system's developing
Personnel Continuity	Very High	Only one person is scheduled for the project

Use of Software Tools	High	A strong use of software tools to handle the project lifecycle is supposed
Multisite Development	Nominal	Average ability of distributed software development
Required Development Schedule	High	Strong constraints imposed on the software development project team

1.3.4 *COCOMO II Analysis result*

Final COCOMO II results using the parameters stated above as input are showed below

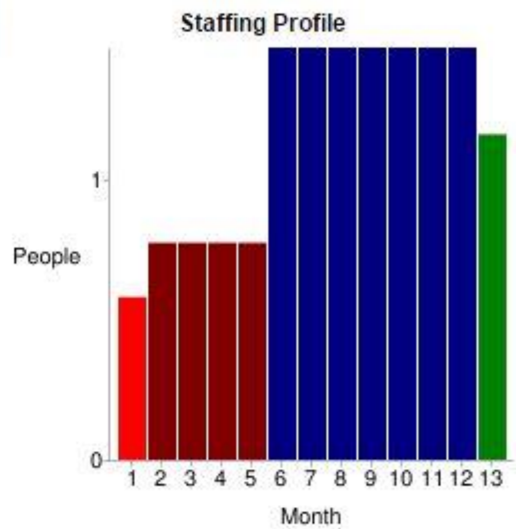
Software Development (Elaboration and Construction)

Effort = 13.6 Person-months
Schedule = 11.3 Months
Cost = $34119

Total Equivalent Size = 4094 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	0.8	1.4	0.6	$2047
Elaboration	3.3	4.2	0.8	$8189
Construction	10.4	7.1	1.5	$25931
Transition	1.6	1.4	1.2	$4094



Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.1	0.4	1.0	0.2
Environment/CM	0.1	0.3	0.5	0.1
Requirements	0.3	0.6	0.8	0.1
Design	0.2	1.2	1.7	0.1
Implementation	0.1	0.4	3.5	0.3
Assessment	0.1	0.3	2.5	0.4
Deployment	0.0	0.1	0.3	0.5

Where:

- I planned an employee is paid 2500$ a month
- Total people needed for the developing of the project can be obtained as follows:

$$People = \frac{Effort}{Duration} = \frac{13,6}{11,3} \cong 1,20 \approx 1$$

2. Project Tasks and Scheduling

Project development isdivided into a set of tasks to be accomplished all over the project lifecycle.

Tasks identification and correspondent schedules, split up in all respective major sub-tasks, are stated in detail in the following table.

Task	**Assignment Date (dd/mm/yyyy)**	**Deadline**	**Detailed Time Schedule**	
Group Constitution	7/10/2015	14/10/2015	Evaluation of the choice for group constitution assignment and submit of the decision on the professor's form	**From:** 07/10/2015 **To:** 13/10/2015 **Total time** = 5 h
			Total task time = 5 h	
***Requirements Analysis and Specifications Document* (RASD) creation**	15/10/2015	06/11/2015	Requirements Elicitation and engineering	**From:** 15/10/2015 **To:** 20/10/2015 **Total time** = 30 h
			Diagrams creation and integration	**From:** 21/10/2015 **To:** 25/10/2015 **Total time** = 25 h
			User interface choices and mockups creation	**From:** 26/10/2015 **To:** 28/10/2015 **Total time** = 10 h
			Alloy system modeling	**From:** 29/10/2015 **To:** 05/11/2015 **Total time** = 23 h
			Final check of the whole document and adjustments	**From:** 06/11/2015 **To:** 06/11/2015 **Total time** = 2 h
			Total task time = 90 h	
***Design Document* (DD) creation**	15/10/2015	04/12/2015	Pondering architecture and design styles and choosing the one to use for the project on the base of elicited	**From:** 09/11/2015 **To:** 15/11/2015 **Total time** = 5 h

			requirements	
			Drafting of the system architecture on the basic ideas of the architecture chosen in previous phase	**From:** 16/11/2015 **To:** 23/11/2015 **Total time =** 25 h
			Diagrams creation and integration	**From:** 24/11/2015 **To:** 30/11/2015 **Total time =** 15 h
			Requirements traceability, final check of the whole document and adjustments	**From:** 01/12/2015 **To:** 04/12/2015 **Total time =** 5 h
			Total task time = 50 h	
Implementation Phase	05/12/2015	08/01/2016	Central Node component and respective subcomponents development and implementation	**From:** 05/12/2015 **To:** 9/12/2015 **Total time =** 25 h
			Central Node component and respective subcomponents unit testing	**From:** 10/12/2015 **To:** 11/12/2015 **Total time =** 10 h
			Zone Server component and respective subcomponents development and implementation	**From:** 12/12/2015 **To:** 17/12/2015 **Total time =** 25 h
			Zone Server component and respective subcomponents unit testing	**From:** 18/12/2015 **To:** 19/12/2015 **Total time =** 10 h
			Dispatcher component development and implementation	**From:** 20/12/2015 **To:** 24/12/2015 **Total time =** 25 h
			Dispatcher component	**From:** 25/12/2015

			unit testing	**To:** 26/12/2015 **Total time** = 10 h
			Client applications development and implementation	**From:** 27/12/2015 **To:** 01/01/2016 **Total time** = 25 h
			Client applications unit testing	**From:** 02/01/2016 **To:** 03/01/2016 **Total time** =10 h
			Database implementation and population	**From:** 04/01/2016 **To:** 9/01/2016 **Total time** = 10 h
			Total task time = 150 h	
***Code Inspection Document* (CID) creation**	09/01/2016	05/02/2016	Central Node component and respective subcomponents code inspection	**From:** 10/01/2016 **To:** 15/01/2016 **Total time** = 7 h
			Zone Server component and respective subcomponents code inspection	**From:** 16/01/2016 **To:** 25/01/2016 **Total time** = 9 h
			Dispatcher component code inspection	**From:** 26/01/2016 **To:** 29/01/2016 **Total time** = 6 h
			Client applications code inspection	**From:** 30/01/2016 **To:** 04/02/2016 **Total time** = 7 h
			Final check of the whole document and adjustments	**From:** 05/02/2016 **To:** 05/02/2016 **Total time** = 1 h
			Total task time = 30 h	
***Integration Test Plan Document* (ITPD) creation**	07/02/2016	21/02/2016	Pondering integration testing approach on the base of system's design and architecture	**From:** 08/02/2016 **To:** 10/02/2016 **Total time** = 5 h
			Identification of components to be integrated and tested on the base of	**From:** 11/02/2016 **To:** 13/02/2016 **Total time** = 5 h

			system's functionalities and integration testing approach	
			Identification and scheduling of test cases on the base of system's functionalities and integration testing approach	**From:** 14/02/2016 **To:** 18/02/2016 **Total time =** 14 h
			Identification and description of tools to be used	**From:** 19/02/2016 **To:** 21/02/2016 **Total time =** 5 h
			Final check of the whole document and adjustments	**From:** 21/02/2016 **To:** 21/02/2016 **Total time =** 1 h
			Total task time = 30 h	
***Project Plan Document* (PPD) creation**	22/02/2016	03/03/2016	Function Points approach description, application and Function Points calculation	**From:** 23/02/2016 **To:** 25/02/2016 **Total time =** 6 h
			COCOMO approach description, application and calculation	**From:** 26/02/2016 **To:** 28/02/2016 **Total time =** 10 h
			Project Task scheduling valuation and drafting	**From:** 29/02/2016 **To:** 01/03/2016 **Total time =** 7 h
			Risk valuation and Risk Analysis	**From:** 02/03/2016 **To:** 03/03/2016 **Total time =** 6 h
			Final check of the whole document and adjustments	**From:** 03/03/2016 **To:** 03/03/2016 **Total time =** 1 h
			Total task time = 30 h	
Total hours of work = 385 h				

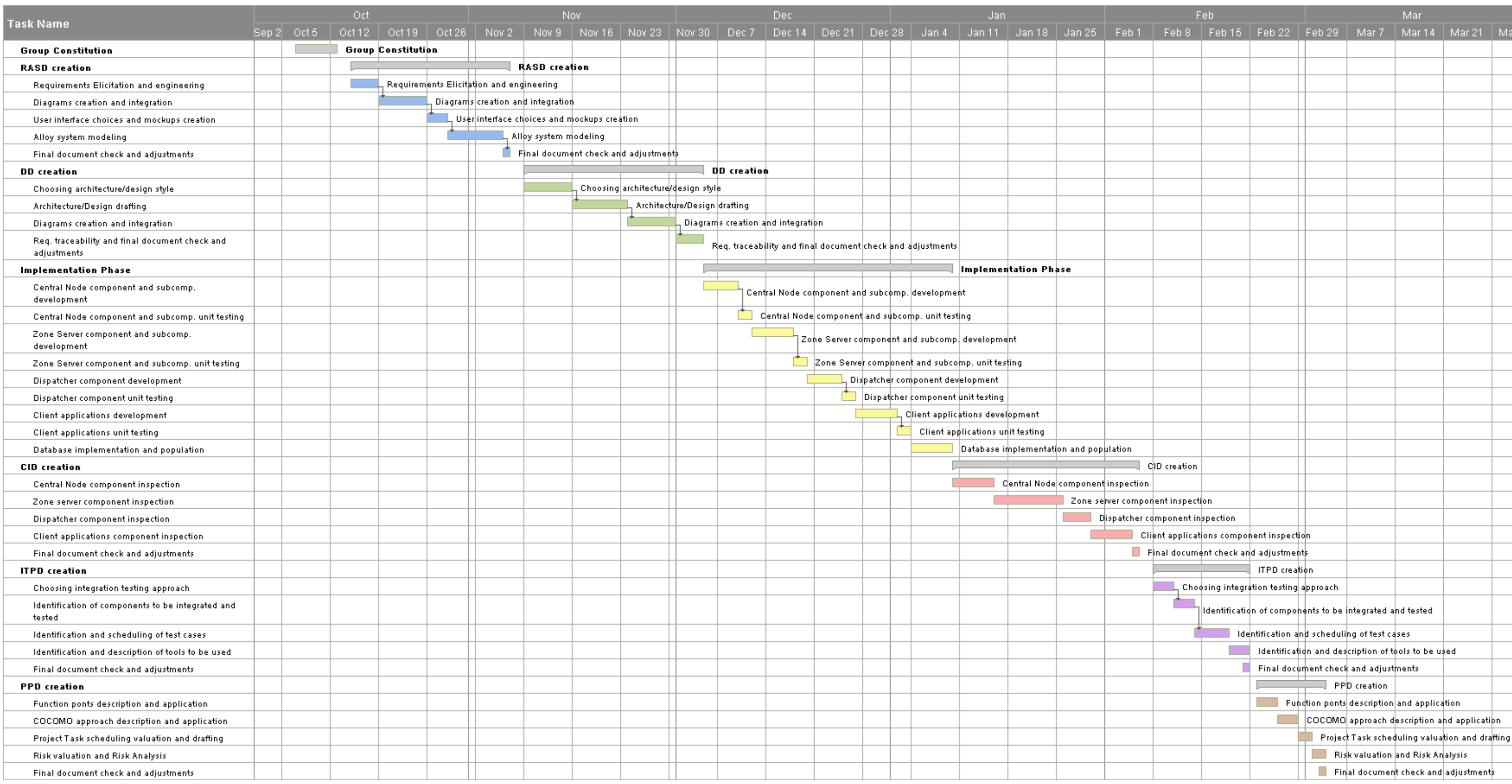
Note I: In the Implementation phase task the use of a third party database application is decide, so the testing phase is skipped.

Note II: Since the time to complete the task is very short, I plan to continue work also on holidays.

Note III: For the work of the drafting of the various parts of the documents to be created at every tasks, is implicitly intended that this work has to be done in parallel which every sub-task (e.g. “Central Node component and respective subcomponents code inspection” → “drafting of code inspection of Central Node component and respective subcomponents code inspection in the Code Inspection Document”

2.1 Gantt Diagram

The following Gantt Diagram shows the task scheduling for the project described in the previous paragraph



Note: arrows in the graph represent task dependencies

3 Resource Allocation

Since the group is formed only of one person, resource allocation doesn't make so much sense, due to the fact that, at any time, the only worker on every part of the project it's me.

4 Risk Analysis

Here risks for myTaxiService project are identified and for each risk a contingency plan is presented.

4.1 Risk Identification

Risk	Probability	Effect (Impact on the system)	Justification for probability and effects
Users aren't able to use the client application due to difficulties in user interface use	Low	Critical	**Probability:** very simple user interface **Effect:** Application cannot be used
Taxi driver aren't able to use the client application due to difficulties in user interface use	Low	Critical	**Probability:** very simple user interface **Effect:** Application cannot be used
Number of users/taxi drivers increases too much and the system can't handle them	Moderate	Critical	**Probability:** If the application becomes very popular number of users/taxi driver can

all anymore			increase a lot **Effect:** Application isn't used
There's a low demand for the use of the application by users (market)	Moderate	Critical	**Probability:** The City could think to improve its public transport service and/or a concurrent company could offer the same, and maybe better, service **Effect:** Application isn't used
Financial problems forces reductions to the project budget	Low	Catastrophic	**Probability:** Budget amount is agreed and reserved at the beginning of the project, it isn't likely to change **Effect:** Application cannot be developed
Worker is ill at critical times in the project	Moderate	Critical	**Probability:** Illness happens, and could happen at any moment **Effect:** Development of the application could be delayed and deadlines could not be met
Changes to requirements that require major design rework **are proposed**	Moderate	Critical	**Probability:** The application has a moderate dimension, so changes can happen **Effect:** Development of some parts of the application could need to be

			completely redesigned, there could be delays and deadlines could not be met
Faults in reusable software components is discovered	Moderate	Critical	**Probability:** The application has a moderate dimension, so reusable software components failure could happen **Effect:** there could be delays due to components repairing and deadlines could not be met

4.2 Risk Management Strategies

Risk	Strategy
User aren't able to use the application	• Prepare a well documented and formed guide to be inserted in any client application • Prepare tutorials to be deployed to users
Taxi driver aren't able to use the client application due to difficulties in user interface use	• Prepare a well documented and formed guide to be inserted in any client application • Prepare tutorials to be deployed to taxi driver: for the deployment, company can be contacted too
Number of users/taxi drivers increases too much and the system can't handle them all anymore	Investigate the possibility to use higher performance hardware components
There's a low demand for the use of the application by users (market)	Increase advertising about the application and consider the possibility to make a deal with a bigger software development house to get advertising about the system inside their applications
Financial problems forces reductions to the project budget	Prepare a document for senior management which shows how the project is contributing to achievement of business goals of the company try to convince them to not cut the budget
Worker is ill at critical times in the project	Organize regular meetings, all over the project lifecycle, with trusted people who could able to work to the project. Give them information which will permit they to proceed on the work in case of illness of the official worker
Changes to requirements that require major design rework are proposed	Get requirements easily traceable to assess their change impact, moreover maximize information hiding in the design.
Faults in reusable software components is discovered	try to replace components which could potentially fail, with bought-in components of known reliability. If there's not enough budget, try to calculate average required time for repairing and try to readapt task scheduling to include this time

5. Appendix

For the creation of this document I spent 30 hours.