2.2 Cost and effort estimation: COCOMO II

In this section we're going to use the COCOMO II approach to estimate the cost and effort needed to develop myTaxiService.

2.2.1 Scale Drivers

Exponent E is the aggregation of five scale factors that account for some possible overheads encountered for software projects. Each factor has a range of rating levels, from Very low to Extra high , and each rating level has a weight. The specific value of the weight is called a scale factor (SF).

In order to evaluate the values of the scale drivers, we refer to the following official COCOMO II table:

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprece-dented	largely unprece-dented	somewhat unprece-dented	generally familiar	largely familiar	thoroughly familiar
SFj	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
SFj	5.07	4.05	3.04	2.03	1.01	0.00
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
SFj	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
SFj	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	Level 1 Lower	Level 1 Upper	Level 2	Level 3	Level 4	Level 5
SF_j	7.80	6.24	4.68	3.12	1.56	0.00

Precedentedness: 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 is LOW because we have some experience of software design but most of the notions used in this project are new to us.

Development flexibility: reflects the degree of flexibility in the development process. Very low means a prescribed process is used; Extra high means that the client only sets general goals. We set it to NOMINAL because we have to follow a prescribed process, but we had a certain degree of flexibility in the definition of the requirements and in the

design process.

Risk resolution: reflects the extent of risk analysis carried out. Very low means little analysis, Extra high means a complete a thorough risk analysis. We set it to HIGH, because many reason (a rather detailed risk analysis is carried out in chapter 5).

Team cohesion: reflects how well the development team know each other and work together. Very low means very difficult interactions, Very high means an integrated and effective team with no communication problems. We set it to VERY HIGH, since the cohesion among the three of us is optimal.

Process maturity: although we had some problems during the development of the project, the goals have been successfully achieved. Since this is our first project of this kind, this value is set to HIGH (Level 3).

The results of our evaluation is the following:

Scale Driver	Factor	Value
Precedentedness (PREC)	LOW	4.96
Development exibility (FLEX)	NOMINAL	3.04
Risk resolution (RESL)	HIGH	2.83
Team cohesion (TEAM)	VERY HIGH	1.10
Process maturity (PMAT)	HIGH	3.12
Total		1.0605

2.2.2 Cost Drivers

The parameter EAF is derived from the effort multipliers (EM) of the Cost drivers. Cost drivers are used to capture characteristics of the product under development, of the personnel working on it, and of general practices that affect the effort to complete the project.

Required Software Reliability:
This is the measure of the extent to which the software must perform its intended function over a period of time. If the effect of a software failure is only slight inconvenience then reliability is low. If a failure would risk human life then reliability is very high.
Our value will be LOW because a failure don't have critical consequences.

RELY Cost Drivers						
RELY Descriptors	slightly inconvenience	easily recoverable losses	moderate recoverable losses	high financial loss	risk to human life	

RELY Cost Drivers						
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	0.82	0.92	1.00	1.10	1.26	n/a

Database size:
When you design a database, you may have to estimate how large the database will be when filled with data. Estimating the size of the database can help you determine the hardware configuration you will require to do the following: Achieve the performance required by your applications. Guarantee the appropriate physical amount of disk space required to store the data and indexes. Estimating the size of a database can also help you determine whether the database design needs refining. For example, you may determine that the estimated size of the database is too large to implement in your organization and that more normalization is required. Conversely, the estimated size may be smaller than expected. This would allow you to denormalize the database to improve query performance. At this stage there is no testing database , but considering the size of the project the DATA cost driver being HIGH.

DATA Cost Drivers						
DATA Descriptors		(D/P) < 10	10 <= (D/P) <= 100	100 <= D/P <= 1000	D/ P > 1000	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	0.90	1.00	1.14	1.28	n/a

Product complexity:
Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations. After considering the size and nature of PowerEnJoy system, we set this parameter to HIGH.

CPLX Cost Driver						
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	0.73	0.87	1.00	1.17	1.34	1.74

Required reusability:
This cost driver accounts for the additional effort needed to construct components intended for reuse on the current or future projects. This effort is consumed with creating more generic design of software, more elaborate documentation, and more extensive testing to ensure components are ready for use in other applications. Since we require

extensive documentation and thorough testing, it is reasonable to set this parameter to high but In our case, the reusability requirements are limited in scope to the project itself, so the RUSE cost driver is set to NOMINAL.

RUSE Cost Driver						
RUSE Descriptors	slightly inconvenience	None	Across project	Across program	Across product line	Across multiple product lines
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	0.95	1.00	1.07	1.15	1.24

Documentation match to life-cycle needs:
The rating scale for the DOCU cost driver is evaluated in terms of the suitability of the project's documentation to its lifecycle needs. We believe that documentation is crucial in any project, therefore we set this parameter to HIGH.

DOCU Cost Driver						
DOCU Descriptors	Many lifecycle needs uncovered	Some lifecycle needs uncovered	Rightsized to lifecycle needs	Rightsized to lifecycle needs	Very excessive for lifecycle needs	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	0.82	0.92	1.00	1.10	1.26	n/a

Execution time constraint:
This is a measure of the execution time constraint imposed upon a software system.The expected amount of CPU used by PowerEnJoy system is estimated to be more then the 85% for this reason we decided to set this value HIGH.

TIME Cost Driver						
TIME Descriptors			<= 50% use of available execution time	70% use of available execution time	85% use of available execution time	95% use of available execution time
Rating level	Very low	Low	Nominal	High	Very High	Extra High

TIME Cost Driver						
Effort multipliers	n/a	n/a	1.00	1.11	1.29	1.63

Storage constraint:
This rating represents the degree of main storage constraint imposed on a software system or subsystem. In our project this parameter is not relevant because we will use less the 50% of use of available storage we decided to set it NOMINAL.

STOR Cost Driver						
STOR Descriptors			<= 50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	n/a	1.00	1.05	1.17	1.46

Platform Volatility:
The term platform is used here to mean the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks. Since the platform shouldn't change too often, this value is set to NOMINAL.

PVOL Cost Driver						
PVOL Descriptors		Major change every 12 mo., minor change every 1 mo.	Major: 6mo; minor: 2wk.	Major: 2mo, minor: 1wk	Major: 2wk; minor: 2 days	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	n/a	n/a	1.00	1.05	1.17	1.46

Analyst Capability:
Analysts are personnel who work on requirements and design. The major attributes that should be considered in this rating are analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate. This parameter is set to HIGH , since we dedicated a great effort in analyzing the problem requirements and its potential integration in the real world, and also in designing the whole system.

ACAP Cost Driver						
ACAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.42	1.19	1.00	0.85	0.71	n/a

Programmer Capability:
Evaluation should be based on the capability of the programmers as a team rather than as individuals. Major factors which should be considered in the rating are ability, efficiency and thoroughness, and the ability to communicate and cooperate. Should development be upon us, we set this parameter to HIGH.

PCAP Cost Driver						
PCAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.34	1.15	1.00	0.88	0.76	n/a

Application Experience:
The rating for this cost driver depends on the level of applications experience of the project team developing the software system. The ratings are defined in terms of the project team's equivalent level of experience with this type of application. Considering our education, we can set this parameter to NOMINAL.

APEX Cost Driver						
APEX Descriptors	<= 2 months	6 months	1 year	3 years	6 years	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.22	1.10	1.00	0.88	0.81	n/a

Platform Experience:

The Post-Architecture model broadens the productivity influence of platform experiences, recognizing the importance of understanding the use of more powerful platforms, including more graphic user interface, database, networking, and distributed middleware capabilities. Our value is VERY LOW because of our few experiences

PLEX Cost Driver						
PLEX Descriptors	<= 2 months	6 months	1 year	3 years	6 years	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.19	1.09	1.00	0.91	0.85	n/a

Language and Tool Experience:
This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem. Software development includes the use of tools that perform requirements and design representation and analysis, configuration management, document extraction, library management, program style and formatting, consistency checking, planning and control, etc. In addition to experience in the project's programming language, experience on the project's supporting tool set also affects development effort. Considering our education, we can set this parameter to NOMINAL.

LTEX Cost Driver						
LTEX Descriptors	<= 2 months	6 months	1 year	3 years	6 years	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.20	1.09	1.00	0.91	0.84	n/a

Personnel continuity:
The rating scale for this factor is in terms of the project's annual personnel turnover. This parameter is set to VERY HIGH, since in our case the available time is less than 6% of the year.

PCON Cost Driver						
PCON Descriptors	48% / year	24% / year	12% / year	6% / year	3% / year	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.29	1.12	1.00	0.90	0.81	n/a

Usage of Software Tools:
This parameter evaluates the use of tools to support the development and testing. Since we set no mandatory prescription to the developers, we set this parameter to NOMINAL.

TOOL Cost Driver						
TOOL Descriptors	edit, code, debug	simple, frontend, backend CASE, little integration	basic life-cycle tools, moderately integrated	strong, mature lifecycle tools, moderately integrated	strong, mature, proactive lifecycle tools, well integrated with processes, methods, reuse	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.17	1.09	1.00	0.90	0.78	n/a

Multisite development:
Given the increasing frequency of multisite developments, and indications that multisite development effects are significant, this parameter measures the impact on the development process of site collocation and communication support. Since we met daily and used phone calls and messaging applications to communicate, we set this parameter to EXTRA HIGH.

SITE Cost Driver						
SITE Collocation Descriptors	Interna-tional	Multicity and multi-company	Multicity or multi-company	Same city or metro area	Same building or complex	Fully collocated
SITE Communications Descriptors	Some phone, mail	Individual phone, fax	Narrow band email	Wideband electronic communication	Wideband elect. comm., occasional video conf.	Interactive multimedia
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.22	1.09	1.00	0.93	0.86	0.80

Required development schedule:

This rating measures the schedule constraint imposed on the project team developing the software. The ratings are defined in terms of the percentage of schedule stretch-out or acceleration with respect to a nominal schedule for a project requiring a given amount of effort. In spite of the well defined deadlines, which facilitated the distribution of our efforts over the time, some phases required much work. For this reason this parameter is set to HIGH.

SCED Cost Driver						
SCED Descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	
Rating level	Very low	Low	Nominal	High	Very High	Extra High
Effort multipliers	1.43	1.14	1.00	1.00	1.00	n/a

Overall, our results are expressed by the following table:

Cost Driver	Factor	Value
Required Software Reliability (RELY)	LOW	0.92
Database size (DATA)	HIGH	1.14
Product complexity (CPLX)	HIGH	1.17
Required Reusability (RUSE)	NOMINAL	1.00
Documentation match to life-cycle needs (DOCU)	HIGH	1.10
Execution Time Constraint (TIME)	HIGH	1.29
Main storage constraint (STOR)	NOMINAL	1.00
Platform volatility (PVOL)	NOMINAL	1.00
Analyst capability (ACAP)	HIGH	0.85
Programmer capability (PCAP)	HIGH	0.88
Application Experience (APEX)	NOMINAL	1.00
Platform Experience (PLEX)	VERY LOW	1.19
Language and Tool Experience (LTEX)	NOMINAL	1.00
Personnel continuity (PCON)	VERY LOW	1.20

Cost Driver	Factor	Value
Usage of Software Tools (TOOL)	NOMINAL	1.00
Multisite development (SITE)	EXTRA HIGH	0.80
Required development schedule (SCED)	HIGH	1.00
Total		1,48792

2.2.3 Effort equation

Sizing the project

A reasonable size estimate of a project is very important for a good model estimation. However, at this stage this is a challenging operation, since it is nearly impossible to correctly guess how much of PoweEnJoY code will be newly written, and how much, instead, will be reused or readapted. That is why we are going to provide the (very unlikely) worst case estimation: we assume that the whole code will be written from scratch. The quantity s is the estimated number of source lines of code (SLOC):

$s = p * FP$

In our case the sizing of the project results:
s= 46* 173=7958 (for lower bound the one we choose to evaluate our project in the worst case)
s= 67* 173= 11591 (for upper bound)

Effort estimation

Effort is expressed in terms of person-months (PM). A person-month is the amount of time one person spends working on the software development project for one month. COCOMO defines the following formula to estimate it:

$e = 2.94 * (s/1000)^E * EAF$

Exponent E and parameter EAF are defined as follows:

$E = 0.91 + 0.01 * \sum(\text{da 1 a 5})Sfi$

$EAF = \prod(\text{i a n})\ EM_i$

In the previous paragraphs we presented the factors from which exponent E and parameter EAF are derived.
Thanks to the results in equations E and EAF, we are now able to estimate the value of the effort, defined in equation e.

E= 1.0605
EAF= 1,48792
e= 2.94 * (7958/1000) ^ 1.06 * 1,48792 =26,04

So, according to COCOMO II model and our estimation, almost exactly 27 person-months are needed to fully develop PowerEnJoy system.

2.2.4 Schedule estimation

Schedule estimation

Now, thanks to the previous results, it is possible to evaluate the time to develop (t). Time to develop is the calendar time in months that goes from the determination of the product's requirements to the completion of an acceptance activity certifying that the product satisfies its requirements.
The time to develop is given by this formula:

t = 3.67 × (e)^F

Exponent F is defined as follows:

F = 0.28 + 0.2 *(E - 0.91)

By substituting the results from previous sections in equations, we obtain:

F= 0.28 + 0.2 (1.0605 - 0.91)= 0.3101
t= 3.67 x (26,04)^ 0.3101= 10.08

The development of PowerEnJoy as a whole is expected to take more than 10.08 months.

By dividing the effort e by the time to develop t , we get the estimated number of people needed for the project:

n = e/t =26,04/10.08= 2.58

5 Risk management

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

PROJECT RISKS

Risk	Probability	Effect
Delays	High	Moderate
Lack of communication	Low	Moderate
Lack of experience	Certain	Moderate
Requirements change	Very low	Moderate

Delays over the expected deadlines: the project could require more time than expected. If this happens, we may release a first, incomplete but working version (e.g. without a ….) and build the less essential features later.

Lack of communication among team members: the team often works remotely and this can lead to misunderstandings in fundamental decisions, to conflicts over the division of the work among team members and to conflicts in code versioning. Those difficulties can be overcome by explicitly defining (e.g. for every delivery we previous discuss the tasks of every member of the group according with our capacity in various works and obviously

talking to each other when some member of the group find some difficulties) the responsibilities of each team member, and by writing clear and complete specification and design documents.

Lack of experience in programming with the specific frameworks:
The team has no actual experience in programming using Java EE. This will certainly slow down the development.

Requirements change: the requirements may change during the development in unexpected way. This risk can't be prevented, but can be mitigated by writing reusable and extensible software.

TECHNICAL RISKS

Risk	Probability	Effect
Integration testing failure	Low	High
Downtime	Moderate	High
Scalability issues	Low	Moderate
Spaghetti code	Low	Moderate
Deployment difficulties	Moderate	Moderate
Data loss	Low	Catastrophic
Data leaks	Moderate	Catastrophic

Integration testing failure: after the implementation of some components, they may not pass the integration testing phase: this can require to rewrite large pieces of software. This risk can be mitigated by defining precisely the interfaces between components and subsystems, and by doing integration tests.

Downtime: the system could go down for excessive load, software bugs, hardware failures or power outages. This risk can be mitigated by building redundant systems and placing them in geographically separate data centers and by performing testing at all levels.

Scalability issues: the system could not scale with a large number of users, requiring a major design rework. A possible plan is to use a cloud infrastructure from a third-party provider to host our system.

Spaghetti code: with the growth of the project, the code may become overloaded, badly structured and unreadable. This risk can be mitigated by writing a good Design Document before starting to code, and by performing code inspection periodically.

Deployment difficulties: if cities already have a Car Sharing system, it could be difficult to deploy our system by keeping and migrating the old data. This problem can be solved by hiring professional system integrators, but of course this would increase the costs.

Data loss: data can be lost because of hardware failures, misconfigured software or deliberate attacks. This problem can be prevented by enforcing a reliable backup plan. Backups should be kept in a separate place from the system, and offline.

Data leaks: misconfiguration, software bugs and deliberate attacks can expose the users' personal data. This risk must be prevented by adopting industry security standards, by encrypting communications and by doing regular penetration testing.

ECONOMICAL RISKS

Risk	Probability	Effect
Bankruptcy	Moderate	Catastrophic
Regulation change	Low	Severe
Competitors	Moderate	Severe

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 asking maybe also some incentive by the city for using green car that take care about the environment.

Regulation change: local and State regulators can change the car sharing regulation at any time, and this could have an unpredictable impact on the usage and the market penetration of our software. This risk can be partially avoided by a good feasibility study. This issue could be mitigated by doing lobbying on the political parties of the affected countries (wherever this is legal). On the other part as in PowerEnJoy case we can be helped by local and State regulator as doing more green car that take care of the environment and in this way people can have more benefit using our cars.

Competitors: other companies like Enjoy or car2GO as can build equivalent or better products at a lower price and put PowerEnJoy out of the market. The competitiveness of our product must be continuously enhanced by introducing innovative features or introduce discounts as the stakeholders yet asked to implement.