

Version: 1.0.0

Release date: 03-01-2017

Riccardo Cattaneo 873647

Fabio Chiusano 874294

PowerEnJoy, Project Plan

Table of Contents

1 Introduction 4

1.1 Purpose and Scope 4

1.2 List of Definitions and Abbreviations 4

1.3 List of Reference Documents 5

2 Size estimation: Function Points 6

2.1 Introduction 6

2.2 Function Points Estimation 7

2.2.1 Internal Logic Files 7

2.2.2 External Logic Files 7

2.2.3 External Inputs 8

2.2.4 External Outputs 8

2.2.5 External Inquiries 9

2.3 Total FP number and summary 9

3 Effort and cost estimation: COCOMO II 11

3.1 Introduction 11

3.2 Scale Factors 12

3.3 Cost Drivers 14

3.3.1 Personnel Capability (PERS) 14

3.3.2 Product Reliability and Complexity (RCPX) 15

3.3.3 Developed for Reusability (RUSE) 16

3.3.4 Platform Difficulty (PDIF) 16

3.3.5 Personnel Experience (PREX) 18

3.3.6 Facilities (FCIL) 18

3.3.7 Required Development Schedule (SCED) 20

3.3.8 Summary 20

3.4 Effort Equation 20

3.5 Duration, number of people and cost 21

3.6 COCOMO II online calculation 22

4 Risks and recovery actions 23

4.1 Introduction 23

4.2 Risk Analysis 24

4.3 Recovery Actions 25

5 Project schedule and resource allocation 26

5.1 Introduction 26

5.2 Tasks 26

5.2.1 Requirements Tasks 27

5.2.2 Design Tasks 27

5.2.3 Setup Tasks 27

5.2.4 Coding Tasks 27

5.2.5 Testing Tasks 28

5.2.6 Delivery Tasks 28

5.3 Task dependencies 28

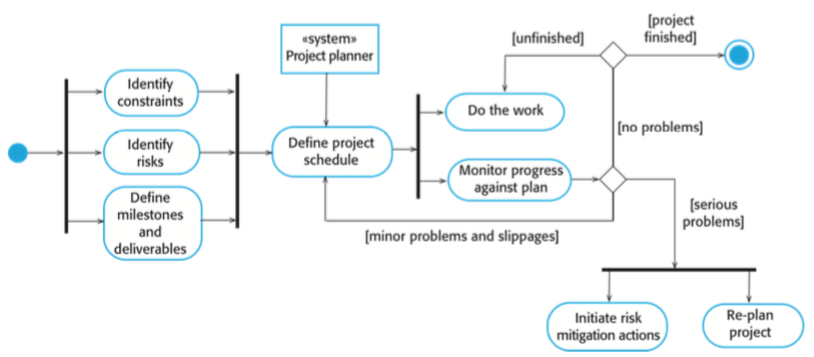
5.4 Resource allocation to tasks and charts 29

6 Effort spent 32

# Introduction

## Purpose and Scope

Project planning is an iterative process that starts in the project startup phase. Plan changes are inevitable as more information about the system to build become available during the development and business goals change.

This is an overview of the project planning process:

This Project Plan Document aims at:

* Estimating the overall project size;
* Thus, estimating the effort needed and the time required to project, develop and test the PowerEnjoy System;
* Identifying all the possible risks, and the related actions that should be taken;
* Defining project schedule and the division of tasks among team members.

## List of Definitions and Abbreviations

* BL: Business Logic;
* ITPD: Integration Test Plan Document;
* DD: Design document;
* RASD: Requirements Analysis and Specification Document;
* COCOMO II: COnstructive COst MOdel, version II.2000.0;
* KSLOC: Kilo Source Line Of Code;
* FP: Function Point;
* UFP: Unadjusted Functional Point;
* ILF: Internal Logic File;
* ELF: External Logic File;
* PM: Person Month;
* Project task: activity that must be completed in order to fulfill the project;

## List of Reference Documents

* Specification Document: Assignments AA 2016-2017.pdf;
* RASD Document v1.1;
* Design Document v1.1;
* ITPD Document v1.0;
* Example of Project Reporting Documents from previous years;

# Size estimation: Function Points

## Introduction

We use the Functional Point approach in order to assess the final dimension of the project. It is based on the evaluation of the functionalities of the software to be, that have been taken from the RASD Document. It’s possible to convert Function Points to KSLOC (i.e. thousands of lines of source code), which will be used in COCOMO to evaluate the effort needed to accomplish the system.

Function Points evaluations depend on the estimator and therefore they are very subjective. Therefore, automatic FP counting is impossible and the results must be accepted with their uncertainties.

The FP types are:

* Internal Logic File: it represents a set of homogeneous data handled by the system;
* External Interface File: it represents a set of homogeneous data used by the application but handled by external application;
* External Input: elementary operation that allows input of data in the system;
* External Output: elementary operation that creates a bitstream towards the outside of the application;
* External Inquiry: elementary operation that involves input and output operations.

The following table outlines the types of Functional Points along with their weight, according to their complexity. The total number of FPs can be computed as the weighted sum of function types using the coefficient listed below:

|  |  |  |  |
| --- | --- | --- | --- |
| *Function Type* | *Complexity* | | |
| *Simple* | *Medium* | *Complex* |
| Internal Logic File | 7 | 10 | 15 |
| External Logic File | 5 | 7 | 10 |
| External Input | 3 | 4 | 6 |
| External Output | 4 | 5 | 7 |
| External Inquiry | 3 | 4 | 6 |

## Function Points Estimation

### Internal Logic Files

The system includes a number of ILFs that will be used to store the information about:

* Users: few simple information, such username, password, email;
* Cars: static information such as model of the car and seats, dynamic information such as battery level and availability. Because of the communication and the synchronization with the remote system of the car, we have considered this point as high complexity;
* Reservations: the system has to keep updated information about each reservation, with related start time and start area, car and user involved, and eventually final cost, obtained with the help of data from the car;
* Rides: information about starting and releasing time of a ride and release area;
* Safe Areas: information on location of a safe area, possibly the possibility of charging a car.

|  |  |  |
| --- | --- | --- |
| *ILF* | *Complexity* | *FPs* |
| User | Low | 7 |
| Car | High | 15 |
| Reservation | Medium | 10 |
| Ride | Medium | 10 |
| Safe Area | Low | 7 |
| Total | | 49 |

### External Logic Files

The system has to interact with two external handlers:

* Payment Handler: the interaction is simple, because the more complex operations are all performed by the external handler, the system has only to receive and store information;
* Search-on-a-map Handler: interaction is more frequent and complex, the system possibly overlays information on received map;

|  |  |  |
| --- | --- | --- |
| *ELF* | *Complexity* | *FPs* |
| Payment | Medium | 7 |
| Map | Medium | 7 |
| Total | | 14 |

### External Inputs

The system interacts with the user to allow him/her to:

* Login/logout: these are simple operations, so we can adopt the simple weight for them;
* Sign-up: validation of payment is performed of external handler, we can adopt the simple weight for this operation;
* Change Payment Info: as above, we have considered it as a simple operation;
* Search for an address: the user insert an address manually and the system shows a map centred on that address. Both the manual insertion and the GPS position are managed maintaining simple interaction between components;
* Search of near available Cars: the user can search for Safe Areas with almost one available car that are near a certain address. The process of this request involves fairly complex algorithm in order to optimize the search performance;
* Search of near safe areas: the user can search for Safe Areas and Special Safe Areas;
* Reserve an available car: The user select one car to be reserved among the available ones. This operation implies multiple interaction: for example, the car has to be marked as unavailable, a reservation code has to be shown in the car dashboard and the expiration of the reservation has to be checked;
* Unlock a car: the user insert the code shown on the car dashboard in order to unlock the car he/she has just reserved, we have considered it as medium complexity because it implies synchronization between multiple components.

|  |  |  |
| --- | --- | --- |
| *Ext. Input* | *Complexity* | *FPs* |
| Login/logout | Low | 2\*3 |
| Sign-up | Low | 3 |
| Change Payment Info | Low | 3 |
| Search for an address | Medium | 4 |
| Search near cars | High | 6 |
| Search near Safe Areas | Medium | 4 |
| Reserve an available car | High | 6 |
| Unlock a car | Medium | 4 |
| Total | | 36 |

### External Outputs

* Ride Conclusion: after that the user has concluded a ride, the system updates ride and reservation entities and calculates the final discharged cost that the user has automatically payed. These data are saved and stored in order to allow the user to consult order history;
* Reservation expired: if the user doesn’t pick up the reserved car in time, the system should mark the reservation as expired, apply the expiration fee and make the reserved car available again.

|  |  |  |
| --- | --- | --- |
| *Ext. Output* | *Complexity* | *FPs* |
| Ride Conclusion | Medium | 5 |
| Reservation expired | Medium | 5 |
| Total | | 10 |

### External Inquiries

* Reservation History: each user can consult his/her personal reservation history. This process does not implies complex algorithms;
* Retrieve Safe Area info: after the user has selected a safe area, the system shows information about available cars in that area and possibly power grids status. Since it implies interaction of multiple entities, we have considered it as medium complexity;
* Show personal info: each user can consult his/her personal info (i.e. his/her information stored in the PowerEnjoy system).

|  |  |  |
| --- | --- | --- |
| *Ext. Inquiry* | *Complexity* | *FPs* |
| Reservation History | Low | 3 |
| Retrieve Safe Area info | Low | 3 |
| Show Personal info | Low | 3 |
| Total | | 9 |

## Total FP number and summary

|  |  |
| --- | --- |
| *Function Type* | *Unadjusted Functional Points* |
| Internal Logic File | 49 |
| External Logic File | 14 |
| External Input | 36 |
| External Output | 10 |
| External Inquiry | 9 |
| Total | 118 |

By summing up all these numerical values we get a total estimation of 118 UFP.

It is possible to estimate a number of line of code from the number of functional points. We have used the value present in the table found at this URL: <http://www.qsm.com/resources/function-point-languages-table>

***[SLOC/UFP] J2EE: 46***

The total estimated number of KSLOC is:

KSLOC/UFP \* UFP = 0.046 \* 118 = **5.428** **KSLOC**

# Effort and cost estimation: COCOMO II

## Introduction

Thanks to COCOMO, we can estimate the amount of effort needed to accomplish the finished system.

This estimation, whose unit of measure in the Person Month, is achieved through a complex, non-linear model that takes in account the characteristics of the product but also of people and process.

The general equation that calculates the effort needed is:

**EFFORT = A \* SIZE E \* EAF**,

where:

* **EFFORT** stands for the estimated number of Person Month for the project, that is the estimated project effort;
* **A** is an organisation-dependent constant. We’ll use the value **2.94**, as suggested by COCOMO II.2000;
* **SIZE** is the project KSLOC. We’ll use the size estimated previously with FP, which is **5.428 KSLOC**;
* **E** reflects the disproportionate effort for large project and is derived from Scale Factors;
* **EAF** (Effort Adjustment Factor) is the product of all the Cost Drivers.

Scale Factors and Cost Drivers are defined in the following sections.

The following diagram summarizes the steps for estimating the effort:

Calculate E

Calculate EFFORT

Calculate EAF

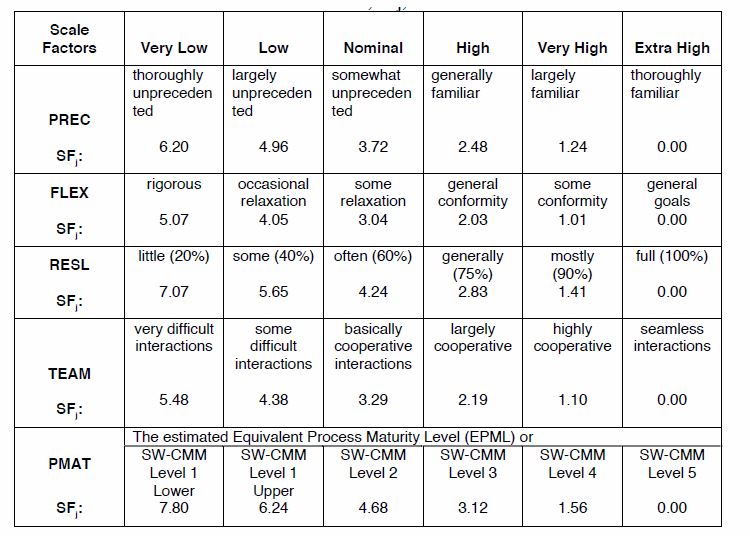
Evaluate Cost Drivers

Evaluate Scale Factors

All the tables used in this analysis have been taken from COCOMO II, Model Definition Manual at:

<http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf>

## Scale Factors



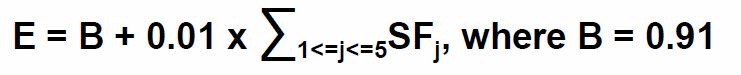
These are the five Scale Factors with our evaluation for this project:

* Precedentedness: It reflects the previous experience that we have with this kind of projects. We have different past experience, but generally this is the first experience using this framework and these development methodologies, this value will be Low;
* Development Flexibility: It reflects the degree of flexibility in the development process. The client has only set some clear general goals, so it is possible to classify the factor as High;
* Risk Resolution: Reflects the extent of risk analysis carried out. Thanks to Risk and Recovery Actions analysis carried out in the following chapters of this document, this factor can be considered as High;
* Team Cohesion: Team members know each other very well, thus the cooperation can be proactive. All the stakeholders seems to have the same vision, so we have evaluated this factor as High;
* Process Maturity: Reflects the process maturity of the organisation. We have concluded that this factor can be evaluated as Nominal.

Results are summarized in the following table:

|  |  |  |
| --- | --- | --- |
| *Scale Factor* | *Evaluation* | *Value* |
| Precedentedness | Low | 4.96 |
| Development flexibility | High | 2.03 |
| Risk resolution | High | 2.83 |
| Team cohesion | High | 2.19 |
| Process maturity | Nominal | 4.68 |
| Total | | 16.69 |

It is now possible to calculate the parameter E of the Effort Formula, using the following:



,

where:

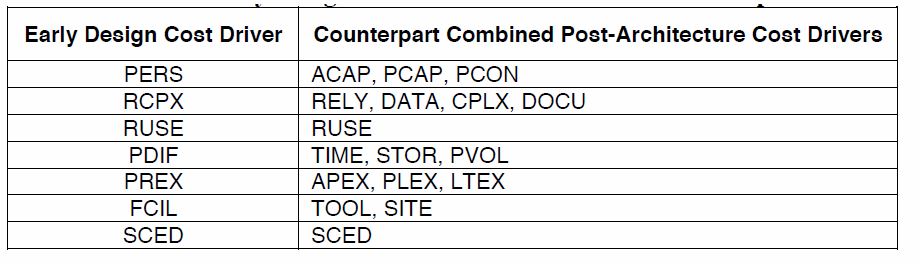
* **B** is a constant whose value is **0.91**, as suggested by COCOMO II.2000;
* **SFj**is the value of the j-th Scale Factor.

Hence **E = 1.0769.**

## Cost Drivers

We have to consider Cost Drivers for an Early Design System, because the PowerEnjoy system has to be built from the ground up.

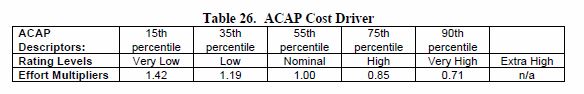
Early design evaluation implies the combination of multiple cost drivers arising from Post Architecture Estimation, following this table:



We will now calculate the value of each early design cost driver by evaluating the corresponding post-architecture cost drivers. Each of these has a rating scale from Very Low (=1) to Very High (=5). Adding up their numerical ratings produces a value, that corresponds to a certain Rating Level with related Effort Multiplier.

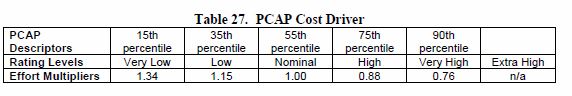
### Personnel Capability (PERS)

* Analyst Capability (ACAP) – High:



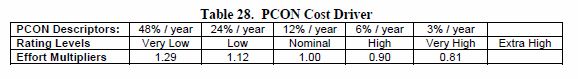
Design and analysis abilities should be set to high, since a lot of effort will be dedicated in analysing the problem specification and requirements, drafting the RASD Document, designing the hardware and software architecture and drafting the Design Document. Moreover, most of the ambiguities present in the initial description will be detailed and resolved in RASD, document that all the stockholders have to approve.

* Programmer Capability (PCAP) – High:

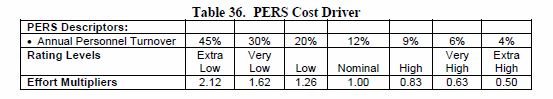


This parameter is evaluated according to our degree of cooperation, which is run in. Thus, the value is set to high.

* Personnel Continuity (PCON) – Low:



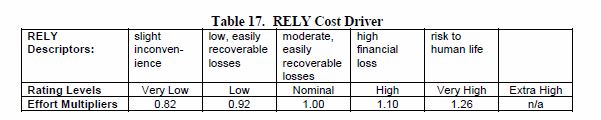
This parameter is relevant in particular since in the current case our available time is less than half a year and we can’t dedicate all our time to the project. For this reason, we have set it to low.

The total numeric value is 4+4+2=10.

This corresponds to a High Rating Level, and its related effort multiplier is **0.83**.

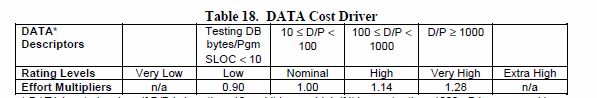
### Product Reliability and Complexity (RCPX)

* Required software reliability (RELY) – Nominal:



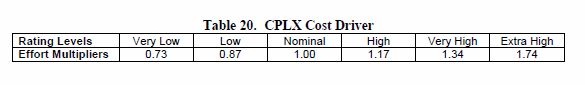
A software failure would not cause risks to human life, but would probably cause some financial problems.

* Database size (DATA) – High:



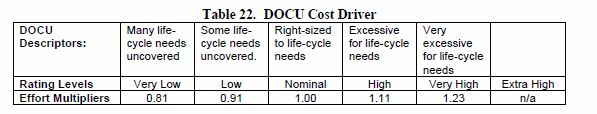
The size of data contained in the database will be probably quite big compared to the number of line of code. Thus, we have set this parameter as High.

* Product complexity (CPLX) – Nominal:

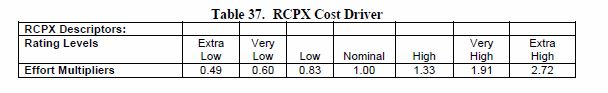


We have evaluated the complexity of control operations, computational operations, device-dependent operations, data management operations and user interface management operations, using the table provided in the COCOMO definition manual. We have evaluated the overall complexity as Nominal.

* Documentation match to life-cycle needs (DOCU) – High:

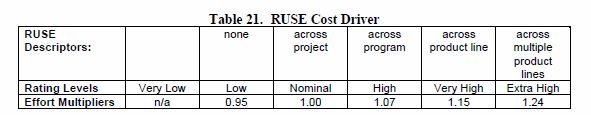


The completeness of documents that will be produced is high, but also the time and efforts required for drawing up them. Although this choice will reduce time and effort needed in other part of the project such as maintenance, because of the better understanding of project and software that all the documents guarantee.

The total numeric value is 3+4+3+4=14.

This corresponds to a High RCPX Rating Level, and its related effort multiplier is **1.33**.

### Developed for Reusability (RUSE)



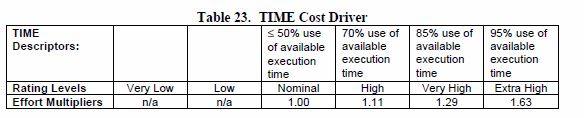
This Early Design model cost driver is the same as its Post-Architecture counterpart.

Thanks to the design that follows common software engineering principles, that enhance the reusability of code, we have set this parameter as High. This value fits also the constraints imposed by the value of Reliability and Database Size.

The High Rating Level corresponds to an effort multiplier of **1.07**.

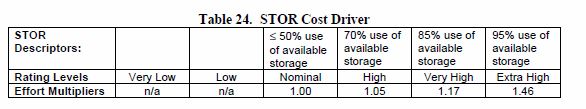
### Platform Difficulty (PDIF)

* Execution time constraint (TIME) – Nominal:



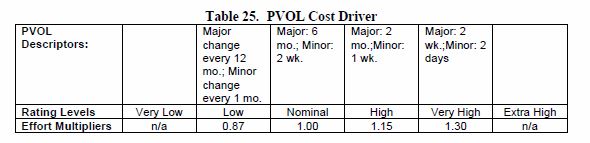
We have evaluated the execution time expected to be used by the system, consuming the total execution time resource, as Nominal. This because the the remarkable increase in available processor execution efficiency of modern CPUs.

* Main storage constraint (STOR) – Nominal:

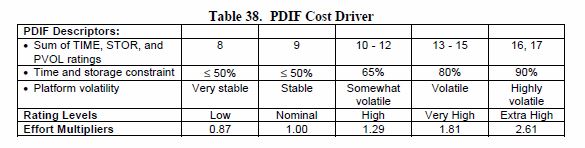


As for the Time constraint, also the storage constraint is subject to the same considerations. Our system has not any particular storage-angry feature, so it is quite simple to get hardware that will not give concerns about space occupation.

* Platform volatility (PVOL) – Nominal:



Changes in Mobile Systems are frequent, anyway a certain degree of backward compatibility is given. Thus, we have set the platform volatility as Nominal.

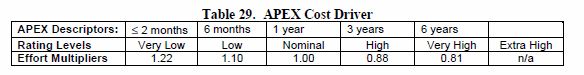


The total numeric value is 3+3+3=9.

This corresponds to a Nominal PDIF Rating Level, and its related effort multiplier is **1.00**.

### Personnel Experience (PREX)

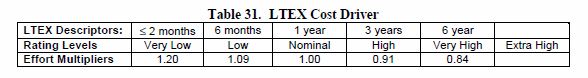
* Application experience (APEX) – Low:



Our project experience is evaluated according to our previous experience in web projects and also according to our abilities in programming in Java and most importantly in the Java EE framework.

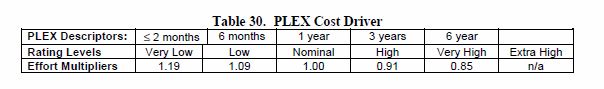
We have different past experiences, but, since we are still students, we can evaluate our experience as a Low value.

* Language and tool experience (LTEX) – Nominal:

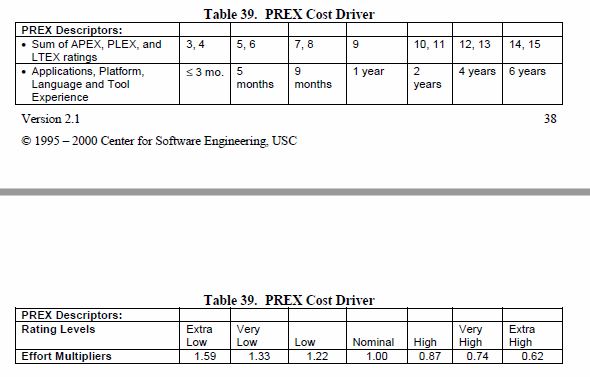


We have started using tools tools involved in software engeneering, such as ones that perform requirements and design representation and analysis, about one years ago. The best level that represents our language and tool experience is Nominal.

* Platform experience (PLEX) - Nominal:



Also our average knowledges about platforms, such as databases, user interfaces and server side development, can be judged as Nominal.

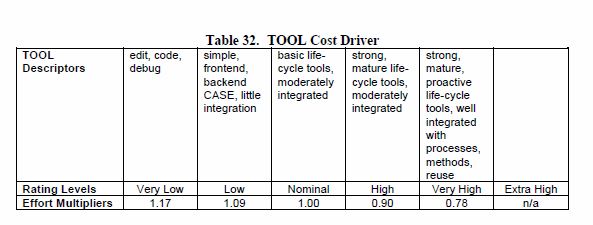


The total numeric value is 2+3+3=8.

This corresponds to a Low PREX Rating Level, and its related effort multiplier is **1.22**.

### Facilities (FCIL)

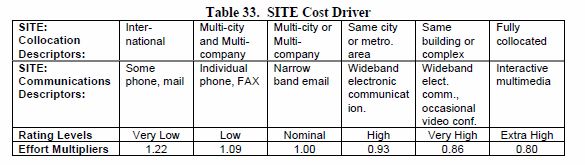
* Use of software tools (TOOL) – Nominal:



Tools that we will use during the project are, for example, Git, NetBeans, Maven and SonarQube, and others that will be mentioned in the other Documents.

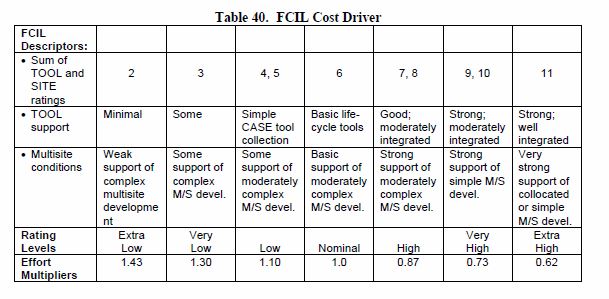
We think that the best level for this cost driver is Nominal, because tools mentioned above are simple basic lifecycle tools that are moderately integrated.

* Multisite development (SITE) – High:



This parameter reflects how we handled the distribution of development over distance and multiple platforms.

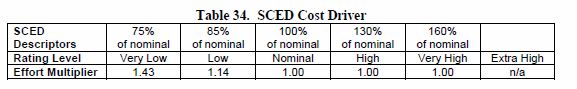
We think that the best value for our situation is High.



The total numeric value is 3+4=7.

This corresponds to a High FCIL Rating Level, and its related effort multiplier is **0.87**.

### Required Development Schedule (SCED)



Also this Early Design model cost driver is the same as its Post-Architecture counterpart.

We plan to distribute our efforts well over the available development time. We have selected the Nominal value for this cost driver.

The Nominal Rating Level corresponds to an effort multiplier of **1.00**.

### Summary

|  |  |  |
| --- | --- | --- |
| *Scale Driver* | *Factor* | *Value* |
| Personnel Capability | High | 0.83 |
| Product Reliability and Complexity | High | 1.33 |
| Developed for Reusability | High | 1.07 |
| Platform Difficulty | Nominal | 1.00 |
| Personnel Experience | Low | 1.22 |
| Facilities | High | 0.87 |
| Required Development Schedule | Nominal | 1.00 |
| Product | | 1.25 |

## Effort Equation

This final equation gives us the effort estimation measured in Person-Months (PM):

**EFFORT = A \* SIZE E \* EAF**,

where:

* **A = 2.94**, as justified in the introduction of the COCOMO chapter;
* **SIZE = 5.428 KSLOC**: estimated lines of code using the FP analysis;
* **E =** **1.0769**:exponent derived from Scale Factors, previously calculated
* **EAF = 1.25**: product of all the Cost Drivers.

With this parameters we can compute the Effort value, that is equal to:

**EFFORT =** 2.94 \* 5.428 1.0769 \* 1.25 **= 22.719 PM**

## Duration, number of people and cost

From the estimated effort, we can deduce the project duration with the following formula:

**DURATION = C \* EFFORTNS (D + 0.2 \* (E - B)) \* (SCED%/100)**,

Where:

* **C** is a constant with value 3.67;
* **EFFORTNS** is the effort excluding the SCED multiplier;
* **D** is a constant with value 0.28;
* **E** is the exponend derived from Scale Factors previously found;
* **B** is a constant with value 0.91;
* **SCED** is a Cost Driver.

Therefore, we have:

**DURATION =** 3.67 \* 22.719 (0.28 + 0.2 \* (1.0769 - 0.91)) = 3.67 \* 22.719 0.31338 = **9.766 M**

Now, from effort and duration, we can deduce the number of people needed to complete the project:

**Npeople = EFFORT / DURATION** = 22.719 / 9.766 = 2.326 = 2 people

Assuming a salary of $1500 per month per person, the total estimated project cost is:

**COST = SALARY \* EFFORT** = 1500 \* 22.719 = **$34078,50**

This is a recap of our estimation procedure:

Cost, people required and time estimation

Effort estimation

Risk management

Size estimation

## Macintosh HD:Users:fabiochiusano:Desktop:magistrale primo anno:Software engineering 2:project:PowerEnjoy:PP:Images:online_1.pngCOCOMO II online calculation

# Macintosh HD:Users:fabiochiusano:Desktop:magistrale primo anno:Software engineering 2:project:PowerEnjoy:PP:Images:online_2.png

# Risks and recovery actions

## Introduction

Risk management is an important phase of project planning, but it is often underestimated. Taking into account risks in advance means being proactive towards them. Primary objective of risk management is to avoid risks and to have a contingency plan in place to handle them in a controlled and effective manner.

There are some principles that we have always to follow:

* Encourage all stakeholders and users to point out risks at any time:

It is important to have an open communication, the whole project can only benefit if no one try to hide information and risks;

* Modify identified risks as more becomes known and add new risks as better insight is achieved;
* Develop a shared product vision, because a shared vision by all stakeholders facilitates better risk identification and assessment.

In next paragraphs, we have analysed some risks, ranking them by probability and impact. We have then developed a contingency plan for those that are not negligible. Therefore, we are using a proactive risk strategy.

## Risk Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ID* | *Type* | *Risk Description* | *Probability* | *Impact* |
| 1 | Project | Project size and complexity grow more than expected | High | Critical |
| 2 | Business | Customer not fully aware of project progress | High | Moderate |
| 3 | Business | Financial problems force reductions in the project budget | Moderate | Critical |
| 4 | Project | The development time is underestimated | Moderate | Critical |
| 5 | Project | Deadlines too close when drafting project documents | Moderate | Critical |
| 6 | Project | Key staff are ill at critical times in the project | Moderate | Serious |
| 7 | Project | Staff experience with used tools is not adequate | Moderate | Serious |
| 8 | Project | Changes to requirements that require major design rework are proposed | Moderate | Serious |
| 9 | Technical | The database used in the system cannot process as many transactions per second as expected | Moderate | Serious |
| 10 | Project | Integration tests shows that design of system has to be improved | Low | Critical |
| 11 | Technical | Car subsystem does not maintain an high level of reliability | Low | Critical |
| 12 | Technical | Database size grows more than expected | Low | Serious |
| 13 | Technical | An external handler changes its interface with the system | Low | Serious |
| 14 | Project | Not complete and correct understanding of requirements | Low | Critical |
| 15 | Techincal | Car internet access not always working, hard to make it reliable | Moderate | Critical |
| 16 | Technical | Bad choiche of car sensors | Low | Serious |

## Recovery Actions

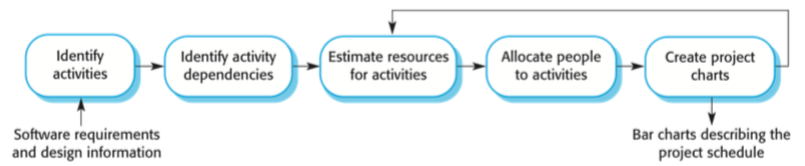
|  |  |
| --- | --- |
| *ID* | *Strategy* |
| 1 | Have a meeting with other team members, where have to be analysed what are the functions, components, interfaces or other piece of software which complexity is higher than expected. Then, check all the possible alternatives that try to simplify the design and the implementation. At the end, a document with all the taken design choices has to be drafted and has to be delivered to all the stakeholders involved. |
| 3 | Draft a briefing document for customers showing progress of the development, showing how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost effective. |
| 4 | Have a meeting with other team members, where point out problems during development that lengthened the schedule. Try to identify tools that would facilitate the development, or try to identify tasks that can be switched between team members. If the situation does not improve, take actions listed for risk 1. |
| 5 | Have a meeting with other team members, where point out what sections of document can be simplified in order to draft in time a simpler, but clear, document. |
| 6 | Try to identify tasks that can be switched between team members, what tasks can be delayed. If the situation does not improve, take actions listed for risk 4. |
| 7 | Have a meeting in which each team member explain what are key features that he knows for each tool. Try to identify documents, material or example on the web that can improve the knowledge. |
| 8 | Alert customer to potential difficulties and the possibility of delays because of changes in already approved RASD; investigate possibly buying-in or simplifying components to be developed. |
| 9 | Investigate the possibility of buying a higher-performance database; investigate the possibility to build more efficient algorithms, which interface with database. |
| 15 | Investigate the possibility to buy better internet modules for the cars; consider to make the cars send data to the server when internet connection is available again. |

# Project schedule and resource allocation

## Introduction

This section describes how the work will be organised as separate tasks and when and how these tasks will be executed.

We should focus on:

* Splitting the project into tasks and estimating time and resources required for each of them;
* Organizing tasks concurrently and creating project charts.

The project scheduling process is summarized in the following diagram:

Some scheduling problems that we must keep in mind are:

* It’s not easy to estimate the difficulty of a problem;
* Productivity is not proportional on the number of workers;
* Adding people to a late project makes the project late;
* Unexpected things always happen.

Therefore, the project development may differ from our schedule. For this reason it is necessary to update the project schedule every certain amount of time to keep it consistent.

## Tasks

In this section the project is breakdown into tasks. Each task is designed so that it should take about a week or two.

The tasks can be divided in the following categories:

* Requirements Taks RT: relative to requirement analysis;
* Design Tasks DT: relative to software design;
* Setup Tasks ST: relative to tier setup (main server setup, car setup, database setup);
* Coding Tasks CT: relative to the project implementation and unit testing;
* Testing Tasks TT: relative to intergation testing;
* Delivery Tasks DT: relative to the delivery of the project.

The dependencies between software components have been already analised in the Integration Testing Plan Document. We can use them to decide the order of software components development and intergation testing, taking advantage of the already found dependencies.

### Requirements Tasks

|  |  |
| --- | --- |
| Task ID | Task description |
| RT1 | Analyse all the requirements and the specifications, make the RASD document |

### Design Tasks

|  |  |
| --- | --- |
| Task ID | Task description |
| DT1 | Study the solutions to our problem, develop the software architecture make the DD document |

### Setup Tasks

|  |  |
| --- | --- |
| Task ID | Task description |
| ST1 | Main server setup (correctly configured as a JEE server, Router class is working), database setup (database running, all tables created) |
| ST2 | Car setup (correctly configured as a JEE server), phones setup (all phones in the equipment required section of the ITPD setup for development) |

### Coding Tasks

|  |  |
| --- | --- |
| Task ID | Task description |
| CT1 | Main server Map Controller |
| CT2 | Main server Payment Controller |
| CT3 | Main server Signup Controller |
| CT4 | Main server Login Controller |
| CT5 | Main server User Controller |
| CT6 | Main server Car Controller |
| CT7 | Main server Search Controller |
| CT8 | Main server Reservation Controller |
| CT9 | Main server Persistence Tier |
| CT10 | Main server JSP |
| CT11 | Main server Web Services |
| CT12 | Car GPS, Seat Sensor, Dashboard Controller |
| CT13 | Car Taximeter Controller, Car Info Controller |
| CT14 | Car PEJControllerCar |
| CT15 | App logic |
| CT16 | App UI |

### Testing Tasks

|  |  |
| --- | --- |
| Task ID | Task description |
| TT1 | Search Controller → Car Controller |
| TT2 | Search Controller → Map Controller |
| TT3 | Signup Controller → Payment Controller |
| TT4 | User Controller → Signup Controller |
| TT5 | User Controller → Login Controller |
| TT6 | Reservation Controller → Car Controller |
| TT7 | Reservation Controller → User Controller |
| TT8 | Reservation Controller → Payment Controller |
| TT9 | PEJControllerCar Controller → GPS Controller,  PEJControllerCar Controller → Seat Sensor Controller,  PEJControllerCar Controller → Dashboard Controller |
| TT10 | PEJControllerCar Controller → Taximeter Controller,  PEJControllerCar Controller → Car Info Controller |
| TT11 | Persistence Manager → Database |
| TT12 | Business Logic Controllers → Persistence Manager |
| TT13 | Business Logic Controllers → Car |
| TT14 | Car→ Business Logic Controllers |
| TT15 | Web Tier → Business Logic Controllers |
| TT16 | Web App → Web Tier |

### Delivery Tasks

|  |  |
| --- | --- |
| Task ID | Task description |
| DelT1 | Deliver product |

## Task dependencies

|  |  |  |  |
| --- | --- | --- | --- |
| Task ID | Effort [person-days] | Duration [days] | Dependencies |
| RT1 | 40 | 30 | - |
| DT1 | 100 | 60 | RT1 |
| ST1 | 2 | 2 | DT1 |
| ST2 | 2 | 2 | DT1 |
| CT1 | 5 | 5 | ST1 |
| CT2 | 5 | 5 | ST1 |
| CT3 | 5 | 5 | CT2 |
| CT4 | 4 | 4 | ST1 |
| CT5 | 10 | 10 | CT3, CT4 |
| CT6 | 10 | 10 | ST1 |
| CT7 | 12 | 12 | CT1, CT6 |
| CT8 | 20 | 14 | CT2, CT5, CT6 |
| CT9 | 7 | 7 | ST1 |
| CT10 | 25 | 17 | ST1 |
| CT11 | 10 | 10 | ST1 |
| CT12 | 10 | 10 | ST2 |
| CT13 | 9 | 9 | ST2 |
| CT14 | 6 | 6 | CT12, CT13 |
| CT15 | 30 | 20 | ST2 |
| CT16 | 30 | 20 | ST2 |
| TT1 | 6 | 6 | CT6, CT7 started |
| TT2 | 7 | 7 | CT1, CT7 started |
| TT3 | 5 | 5 | CT2, CT3 started |
| TT4 | 6 | 6 | CT3, TT3, CT5 started |
| TT5 | 7 | 7 | CT4, CT5 started |
| TT6 | 5 | 5 | CT6, CT8 started |
| TT7 | 7 | 7 | CT5, TT4, TT5, CT8 started |
| TT8 | 7 | 7 | CT2, CT8 started |
| TT9 | 7 | 7 | CT12, CT14 started |
| TT10 | 8 | 8 | CT13, CT14 started |
| TT11 | 8 | 8 | ST2, CT9 started |
| TT12 | 9 | 9 | CT9, TT11, started a business controller |
| TT13 | 10 | 10 | CT8, CT14 |
| TT14 | 10 | 10 | TT12, TT13 |
| TT15 | 20 | 15 | TT12, TT13, TT14 |
| TT16 | 20 | 15 | CT15, CT16, TT15 |
| DelT1 | 10 | 10 | TT16 |

## Resource allocation to tasks and charts

In order to allocate resources to tasks, the best thing to do is to find the critical path in the graph of the task dependences and allocate project members in a way to minimize interructions on the critical path, according to the availability of each project member. We’ve chosen to alternate each other on the critical path.

However, due to the fact that we are just two people, there weren’t a lot of opportunities to parallelize tasks.

We took into account personal availabilities and holidays.

Here’s the Gantt diagram that represents the project schedule along with resource (i.e. project members) allocation to tas

Macintosh HD:Users:fabiochiusano:Desktop:magistrale primo anno:Software engineering 2:project:PowerEnjoy:PP:Images:Gantt.pdf

# Effort spent

We managed to distribute the worklooad fairly between days and team members in a way that allowed us to finish a few days before the deadline and have time for an accurate check in the last days.

The total amount of time required to build this document is about 14 hours for Riccardo Cattaneo and 10 hours for Fabio Chiusano.